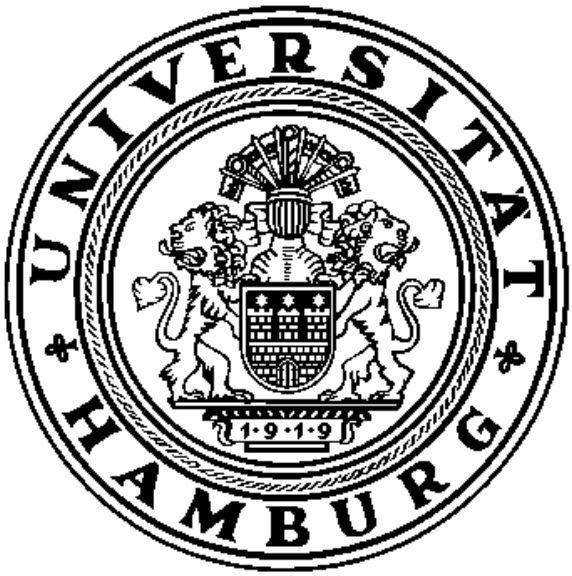
A Hybrid Genetic Swarm Algorithm for Interactive Inverse Kinematics



Sebastian Starke

A thesis submitted for the degree of
Master of Science

Technical Aspects of Multimodal Systems (TAMS)
Knowledge Technology (WTM)

Faculty of Mathematics,
Informatics and Natural Sciences
University of Hamburg

June 27th, 2016

Supervisors: Prof. Dr. Jianwei Zhang
Dr. Sven Magg
Advisor: Dr. Norman Hendrich

Sebastian Starke
Student Number: 6146302
Rübenkamp 86
22307 Hamburg

Abstract

This thesis presents a novel biologically-inspired approach for solving the inverse kinematics problem efficiently on arbitrary joint chains. It provides high accuracy, convincing success rates and is capable of finding suitable solutions in real-time. The algorithm tackles the problem by optimization and incorporates joint constraints, reliably avoids getting stuck in suboptimal extrema and scales notably well even for increasing non-linearity as well as greatly higher-dimensional degree of freedom. It can achieve minimal joint displacement with respect to the amount of change within the target and successfully converges to a solution even when the target objective can not be satisfied by any configuration in joint space.

The algorithm merges the evolutionary and collective strengths of genetic algorithms and swarm intelligence which results in biologically plausible hybridization. Applying randomized multi-objective weights follows the principle of natural evolution within continually changing environments and genetic niching then allows for an efficient simultaneous exploitation of local extrema for multimodal solutions where dead-end paths can be detected by a simple heuristic.

Experimental results show that the presented solution performs significantly more robust and adaptive than traditional or various related methods and can be tuned for individual performance and behaviour preferences which is greatly beneficial for interactive applications. The algorithmic design is modular and generic and thus might also be applied to various other problems that can be solved by optimization techniques.

Zusammenfassung

Diese Arbeit präsentiert einen neuartigen biologisch-inspirierten Ansatz, um das Problem der inversen Kinematik auf beliebigen Gelenkketten effizient zu lösen. Dieser bietet sowohl hohe Genauigkeiten als auch überzeugende Erfolgsraten und ist fähig, geeignete Lösungen in Echtzeit zu finden. Der Algorithmus basiert auf Optimierung und beachtet Einschränkungen der Gelenke, erkennt und vermeidet suboptimale Extrema zuverlässig und besitzt bemerkenswerte Skalierbarkeit unter zunehmender Nichtlinearität sowie enorm hochdimensionalen Freiheitsgraden.

Der Algorithmus vereint die evolutionären und kollektiven Vorteile genetischer Algorithmen und der Schwarmintelligenz, was letztlich in einer biologisch plausiblen Hybridisierung resultiert. Die Verwendung von randomisierten Gewichtungen für eine Optimierung unter mehreren Zielfunktionen verfolgt das Prinzip von natürlicher Evolution innerhalb einer sich kontinuierlich verändernden Umwelt. Genetische Nischenbildung ermöglicht letztlich eine effiziente simultane Verbesserung innerhalb lokaler Extrema, um multimodale Lösungen zu erhalten, wobei nicht zielführende Pfade durch eine simple Heuristik erkannt werden können.

Die Resultate einer experimentellen Analyse zeigen, dass der präsentierte Algorithmus eine signifikant höhere Robustheit sowie Adaptivität gegenüber traditionellen sowie verschiedenen verwandten Ansätzen besitzt. Zudem kann dieser entsprechend individueller Performanzaspekte angepasst werden, was hinsichtlich interaktiver Anwendungen besonders vorteilhaft ist. Das algorithmische Design ist modular sowie generisch und kann demnach ebenfalls auf verschiedene andere Probleme angewandt werden, die durch Optimierungsverfahren gelöst werden können.

Acknowledgements

First and foremost, I would like to greatly thank Prof. Dr. Jianwei Zhang and also Dr. Sven Magg for supervising my thesis and for their infinite patience as well as valueable inspirations, guidance and comments on my work.

I further want to give my sincere thanks to Dr. Norman Hendrich not only for his scientific and helpful advices during this thesis, but also for his outstanding and continuous support with immense knowledge throughout my complete study.

Also, I would like to show my appreciation to Dr. Matthias Kerzel, Dennis Krupke, Lasse Einig, Florens Wasserfall and also to all other members of the TAMS and WTM groups whose office doors were always open for any kind of conversation.

I want to thank my fellow students and good friends Matthis, Jonas and Tobias for the interesting and motivating discussions, sleepless nights before deadlines and the best collaborative preparation for exams I could ever have imagined.

Many thanks also go to my family for their everlasting backing and support that brought me through my study and life in general.

Finally, I must express my deepest love and gratitude in all respects to my wonderful girlfriend Christina. This accomplishment would not have been possible without her. Thank you.

"Nature is the source of all true knowledge. She has her own logic, her own laws, she has no effect without cause nor invention without necessity."

– Leonardo da Vinci

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"It is not the most intellectual or the strongest species that survives, but the species that survives is the one that is able to adapt to or adjust best to the changing environment in which it finds itself." – Charles Darwin

In natural sciences, we are typically concerned with understanding natural phenomena and try to discover appropriate models, systematic descriptions or mathematical formulas to imitate all kinds of simple to complex processes. To any given problem, we design an algorithm by which we intend to provide an optimal solution. We consider all kinds of available information and constraints, integrate knowledge of which we specifically assume to describe our model best – and in the end we fail in genericity and adaptivity simply because we unintentionally overfit by making our approach work best for some certain predefined scenario. This issue becomes even more noticeable by the ever increasing amount of data and requirements to manage high-dimensional information as well as dynamically changing environments.

The goal of computer science and artificial intelligence nowadays is to overcome exactly this kind of problem. Purely mathematical and statistical approaches have proven to perform successfully under constrained and well-defined settings, but standards scale with rising computational power which allows for more advanced and adaptive models and algorithms. In fact, nature itself mostly provides optimal solutions for many problems where it is beneficial to – if possible – adopt the underlying patterns and processes. Gaining more insight into this area of research will help to create more dynamic and flexible solutions and thus allows to enhance intelligence rather than artificiality in computational systems.

This thesis adresses the problem of inverse kinematics which constitutes one of the most fundamental issues in robotic systems and control of motion. It is formulated by attempting to find a suitable configuration of joint variables satisfying a Cartesian objective and represents the opposite to forward kinematics which describes deriving a certain Cartesian result from a given consecutive joint variable configuration. While the computation of forward kinematics is straight-

forward by a direct mathematical mapping from joint to Cartesian space through coordinate transformations and thus analytically solveable, the same does not hold true the other way round. For inverse kinematics, analytical closed-form solutions can only be derived for specific kinematic geometry that is typically correlated to a comparatively low degree of freedom. As soon as the kinematic model becomes more complex, analytical methods are not available and efficient numerical methods are required. More particularly, inverse kinematics represent no bijective mapping but a relation between Cartesian and joint space for which zero up to infinite configurations can exist and it remains difficult to prefer one over another. In addition, several other issues arise concerning singularities, joint constraints and displacement followed by self-collision as well as smooth and collision-free planning of motion, multiple end effectors and lastly incorporation of physical dynamics.

Some of the most widely known and popularly applied traditional methods are based upon computing the *Jacobian* [11]. The solution for inverse kinematics is then either found by taking its transpose [5, 93], pseudoinverse [91, 57], the damped least squares [87, 33] or by further improvements using singular value decomposition [10]. However, problems were not only observed by getting stuck in suboptimal extrema depending on the geometric complexity of the kinematic model but also by unrealistic motion. Constructive approaches successfully tried to overcome a few of these issues by performing random or heuristic restarts from random initial configurations [8]. Further, the *Cyclic Coordinate Descent* [88] method is acknowledged for its simplicity in computation and accuracy but whereby – unfortunately – resembling issues remain as for the *Jacobian*. *FABRIK* [3] outlines a novel efficient iterative geometric approach to handle multiple end effectors and but encounters difficulties when dealing with orientation constraints. Recently, much acceptance has also been paid to algebraic [19, 20] approaches that are convincing due to their extremely low computational cost but encounter problems by highly articulated, constrained and increasingly complex kinematic models and thus lack in scalability. Same drawback holds for solutions that rely on expensive precomputation and reachability analysis [86] which therefore are restricted to a lower-dimensional degree of freedom. Other approaches formulate the problem by biologically-inspired models such as *Particle Swarm Optimization* [74, 73, 24, 90, 72, 15] or *Genetic Algorithms* [76, 77, 63, 1, 84]. Those clearly convince in terms of dimensional scalability, robustness and genericity resulting in less difficulties with highly articulated kinematic models. However, they typically depend on a large number of parameters and might not converge as fast or accurate as other approaches which incorporate information about the gradient or specific geometric properties. Lastly, *Artificial Neural Networks* have been applied to approximate the inverse kinematics function by learning rather than computing the relations between joint and Cartesian space [16, 42, 36, 28]. Nevertheless, the optimal choice of training samples remains unclear, the training must be applied in advance and can take unacceptably long and finally the attainable precision might be too large. All together, this demonstrates the complexity of this ongoing area of research and offers the opportunity for improvements and the design of novel scientific approaches.

1.1 Motivation

Inverse kinematics is a fascinating research problem where numerous sophisticated algorithms have been presented during the last decades but no universal solution could yet be found. Most prominent approaches either attempt to directly derive a geometric solution from the kinematic model or aim to solve the problem iteratively by following the gradient where suboptimal extrema are just one major issue. Others try to interpolate the solution from precomputed joint space configurations under some certain domain-specific resolution but suffer from the *Curse of Dimensionality* [9]. When successfully mastering these issues, approaches may still fail to obtain sufficiently small errors, continue convergence or to succeed within a reasonably short amount of time.

While inverse kinematics has originally evolved from the discipline of robotics, it has lately gained significant importance also in the field of character animation and interaction. Therefore, it is no more only related to industrial robot systems but also to applications settled in the movie and video game industry which represent an increasingly growing market and community. Modern game engines like *Unity3D* [81, 53] and *Unreal Engine* [26, 54] are becoming more popular and greatly ease the creation of virtual reality and the research in human-computer as well as human-robot interaction. Inverse kinematics has become an interdisciplinary study where it is no more solely required to be solved as fast and accurate as possible, but also to obtain plausible solutions on arbitrary kinematic models.

To successfully master these challenges and considering the growing demand in intelligent and adaptive systems, more universal solutions are required in order to create a realistic and interactive behaviour of artificial life that will inevitably be part of the next generation in robotics, virtual reality and modern technology.

1.2 Research Question

The objective of this thesis is to design an efficient biologically-inspired approach to the inverse kinematics problem that is primarily based on genetic algorithms and particle swarm optimization and which can find accurate solutions in real-time even for highly articulated kinematic models. While related methods often encounter problems with the problem of suboptimal extrema, it is valuable to discover new strategies that simultaneously serve exploitation and exploration but remain efficient and sensitive to local extrema. In addition to this, the applicability for industrial robots as well as for interactive applications in virtual reality is of high interest. Ultimately, evolutionary algorithms offer to be revisited from a theoretical point of view with intent to make them behave in a more biologically plausible way. Traditionally, genetic algorithms optimize a problem by random jumps without consideration of system changes and dynamics but what is not true for the principle of natural evolution.

1.3 Outline

While this chapter served for introducing the problem of inverse kinematics from an aerial perspective in order to provide a rough overview, the following chapters are structured as follows:

Chapter 2 first provides the necessary fundamental knowledge for robot kinematics that is required from a mathematical perspective but also introduces important related terms of robot technology. Afterwards, the algorithmic principle of analytical and numerical algorithmic approaches will be discussed leading to the efficient technique of randomization. Lastly, an insight into the area of biologically-inspired artificial intelligence with particular detail on genetic algorithms and particle swarm optimization is given.

Chapter 3 then continues providing a more detailed description and presentation of the current state of the art related to inverse kinematics. More particulary, this includes the class of approaches based on the *Jacobian*, the *Cyclic Coordinate Descent*, *FABRIK*, *Particle Swarm Optimization* as well as *Genetic Algorithms* and finally *Artificial Neural Networks*.

Chapter 4 subsequently presents the algorithmic approach that was developed during this thesis. Introducing with a complete overview on the whole algorithm, the decision of each design aspect, improvement and modification will be explained and discussed in very detail with regard to the initial problem statement.

Chapter 5 follows with an experimental analysis based upon the approach described in chapter 4 and is subdivided into four parts. Those cover an initial parameter study which is then followed by a selective study which highlights the effects of the individual algorithmic improvements. Subsequently, a performance and comparative study on the complete algorithmic solution is conducted on various kinematic models which explicitiely demonstrates the efficiency and flexibility of the algorithm.

Chapter 6 concludes with a summary, a list of contributions and an outlook to future work.

Chapter 2

Fundamental Knowledge

"In order to arrive at knowledge of the motions of birds in the air, it is first necessary to acquire knowledge of the winds, which we will prove by the motions of water in itself, and this knowledge will be a step enabling us to arrive at the knowledge of beings that fly between the air and the wind." – Leonardo da Vinci

Robot kinematics is primarily based on advanced principles in geometric algebra and functional relations. Consequently, section 2.1 first briefly introduces related principle aspects and continues with formal descriptions of coordinate transformations as well as forward and inverse kinematics. Further, various knowledge that covers the degree of freedom of a kinematic model, joint types as well as their constraints and finally the problem of singularities will be presented.

Since chapter 1 has already mentioned that analytical methods may not always be sufficient to solve the problem of inverse kinematics, section 2.2 explains the algorithmic concepts of numerical optimization as well as randomization in order to efficiently approximate to accurate solutions.

Section 2.3 lastly outlines the science field of biologically-inspired artificial intelligence and introduces the concepts of evolutionary and collective systems which consitute a major significance within this thesis.

2.1 Robot Kinematics

Kinematics is a subfield from the domain of *Mechanics* and defines the positional and rotational motion of objects with respect to the physical properties of velocity and acceleration but without consideration of mass, force and torque which are part of *Dynamics*. The fundamental equations of kinematics are given by (2.1) (velocity) and (2.2) (acceleration) where $\mathbf{r}$ is a vector of translation or rotation. [62]

$$\mathbf{v}(t) = \dot{\mathbf{r}}(t) = \frac{d\mathbf{r}(t)}{dt} \tag{2.1}$$

$$\mathbf{a}(t) = \ddot{\mathbf{r}}(t) = \frac{d\mathbf{v}(t)}{dt} \tag{2.2}$$

In *Robotics*, the motion of a *Manipulator* from its *Base* to its *End Effector* is defined by a *Kinematic Chain* which is given by a consecutive set of *Links* and *Joints*. Each joint then defines a particular axis of motion which describes the either translational or rotational movement for the connected link. The hierarchical composition of these elements is related to a sequence of relative successive coordinate transformations which then define the individual aligned coordinate systems. An example of a serial kinematic chain is given by Fig. 2.1. Further, these transformations are most commonly expressed using the *Denavit-Hartenberg* notation which uses homogeneous matrices that can be constrained from a six- to four-dimensional concatenation given their *D-H Parameters*. [17, 18, 46, 56]

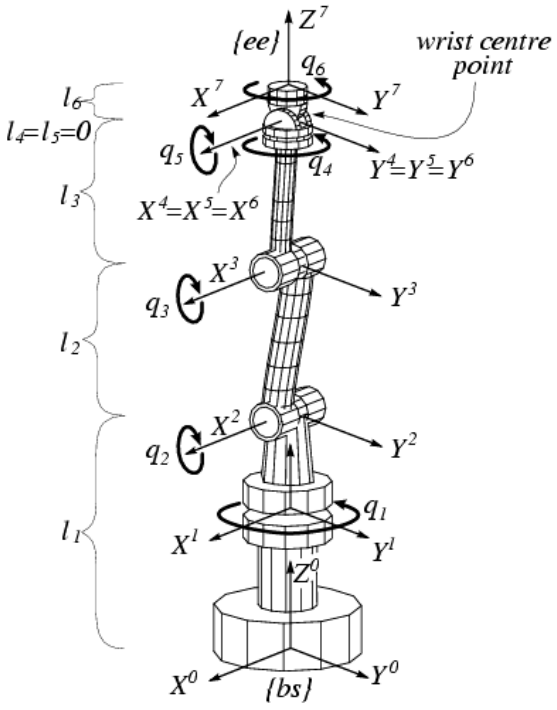


Figure 2.1: Example of a Robotic Kinematic Model [40]

The posture of the manipulator is controlled by a set of joint variables which then allow to compute the forward kinematics or must be set by means of inverse kinematics. Fig 2.2 illustrates the general principle of forward and inverse kinematics on how to obtain a result in Cartesian space from joint space variables $\theta_1, ..., \theta_n$ through applying transformations $T_0, ..., T_n$ or vice versa. [61, 46]

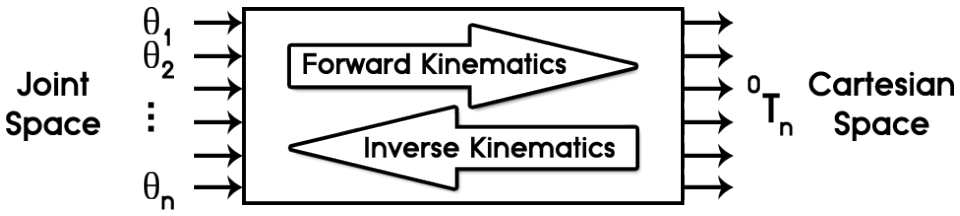


Figure 2.2: Principle of Forward and Inverse Kinematics

2.1.1 Coordinate Transformations

In robot kinematics, geometric coordinate transformations between two domains A, B can be defined through a bijective mapping $f : A \mapsto B \wedge f^{-1} : B \mapsto A$ where $ff^{-1} = I$ and $f : \mathbb{R}^n \mapsto \mathbb{R}^n$. Further, only translation and rotation are required which can be consistenly specified using homogeneous transformation matrices as shown in (2.3) where $t, R \in \mathbb{R}^3$ denote a three-dimensional translation vector or rotation matrix.

$$T_t = \begin{bmatrix} 1 & 0 & 0 & \\ 0 & 1 & 0 & t \\ 0 & 0 & 1 & \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad T_R = \begin{bmatrix} & & & 0 \\ & R & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

The corresponding rotation matrices $R_x, R_y, R_z \in \mathbb{R}^3$ are always constrained to a rotation about a certain axis by an angle α and are denoted by (2.4) considering the right-hand rule where S and C abbreviate the sine and cosine functions.

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & C\alpha & -S\alpha \\ 0 & S\alpha & C\alpha \end{bmatrix} R_y(\alpha) = \begin{bmatrix} C\alpha & 0 & S\alpha \\ 0 & 1 & 0 \\ -S\alpha & 0 & C\alpha \end{bmatrix} R_z(\alpha) = \begin{bmatrix} C\alpha & -S\alpha & 0 \\ S\alpha & C\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

Successive multiplication of transformations $T_0, ..., T_n$ then again gives a transformation matrix where its first three columns define the axis alignments and fourth the position of the final resulting coordinate system. Since matrix multiplications usually do not offer commutativity, it can be shown that left-hand side multiplication causes global transformation while right-hand side multiplication results in a local transformation what is of particular interest for various applications. [51, 23]

Another way to perform coordinate transformations in kinematics and especially in computer graphics is to use vector additions for translation and quaternions for rotation. Given a set of Euler angles (ϕ, θ, ψ) where ϕ denotes a *Roll* about the X-axis, θ a *Pitch* about the Y-axis and ψ a *Yaw* about the Z-axis, homogeneous transformations can be used to determine the orientation of a coordinate system. However, quaternions are from a computational point of view much more efficient and also semantically consistent in contrast to Euler angles. For the latter, the order of rotation matters although quaternion multiplication itself does also not generally provide commutativity. A quaternion is simply a four-dimensional vector defined by (2.5) where the final orientation is determined by rotating about an arbitrarily defined three-dimensional normalized axis (a_x, a_y, a_z) by an angle α.

$$q = (x, y, z, w) = (a_x S\frac{\alpha}{2}, a_y S\frac{\alpha}{2}, a_z S\frac{\alpha}{2}, C\frac{\alpha}{2}) \quad x, y, z, w \in \mathbb{R} \tag{2.5}$$

The position then is obtained by incorporating the internal quaternion rotation to the axis of translation where the rotation can be expressed by (2.6). [51, 23]

$$R_q = \begin{bmatrix} 1 - 2y^2 - 2z^2 & 2xy - 2zw & 2xz + 2yw \\ 2xy + 2zw & 1 - 2x^2 - 2z^2 & 2yz - 2xw \\ 2xz - 2yw & 2yz + 2xw & 1 - 2x^2 - 2y^2 \end{bmatrix} \tag{2.6}$$

2.1.2 Forward Kinematics

The objective of forward kinematics is to determine the configuration $\mathcal{X}$ in Cartesian space from a consecutive set of joint variables $\theta = \theta_{i=1,\dots,n}$ in joint space what can formally be denoted as (2.7).

$$\mathcal{X} = f(\theta) \tag{2.7}$$

The transformation from the base to the end effector link is then given by a series of successively applied transformations (2.8) where ${}^{i-1}T_i$ describes a transformation from link l_{i-1} to l_i.

$${}^{base}T_{ee} = {}^0T_1 \ {}^1T_2 \ \dots \ {}^{n-1}T_n \tag{2.8}$$

Revisiting the previously introduced *Denavit-Hartenberg* notation in section 2.1 as one of the most common techniques to compute the forward kinematics of a robotic manipulator, there are basically three rules to be followed:

- The z_{i-1}-axis is set along the axis of motion of joint i
- The x_i-axis is normal to the z_{i-1} axis and points away from it
- The y_i is selected such that it completes the right-handed coordinate system by the cross product $y_i = z_i \times x_i$

Each transformation ${}^{i-1}T_i$ is defined by composition of four homogeneous transformations (2.9) where its parameters are:

- A rotation θ_i about the z_{i-1} axis
- A translation d_i along the z_{i-1} axis
- A translation a_i along the x_i axis
- A rotation α_i about the x_i axis

$$ {}^{i-1}T_i = T_R(z_{i-1}, \theta_i)T_t(z_{i-1}, d_i)T_t(x_i, a_i)T_R(x_i, \alpha_i) \tag{2.9} $$

Ultimately, the final homogeneous transformation matrix between two consecutive links controlled by their joint variable θ_i results in (2.10). [46, 71, 17, 18]

$$ {}^{i-1}T_i = \begin{bmatrix} C\theta_i & -C\alpha_i S\theta_i & S\theta_i S\alpha_i & a_i C\theta_i \\ S\theta_i & C\alpha_i C\theta_i & -S\alpha_i C\theta_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.10} $$

As observed, the solution to forward kinematics is straightforward and can be efficiently obtained by analytical computation. Since the result is always mathematically accurate and unique, calculating the forward kinematics is often used in order to iteratively approximate a solution in inverse kinematics.

2.1.3 Inverse Kinematics

The problem of inverse kinematics is formulated in reverse to forward kinematics denoted by (2.11) where the particular interest is in determining a joint variable configuration $\theta = \theta_{i=1,\dots,n}$ in joint space that satisfies a given target $\mathcal{X}$ in Cartesian space. This Cartesian objective is mostly given by a six-dimensional pose but can also consist only of a three-dimensional position or orientation depending on the application.

$$ \theta = f^{-1}(\mathcal{X}) \tag{2.11} $$

In contrast to forward kinematics, no general analytical solution is available and zero up to infinite solutions can exist. This already indicates the complexity and ambiguity of inverse kinematics. Although it is possible to derive an algebraic solution for simple kinematic geometry with lower-dimensional degree of freedom, this methods breaks down for increasing geometric complexity so that most approaches rely on numerical approximation to solve an inverse kinematics problem. This technique however brings other issues into play such as suboptimal extrema which correlates to presuming a suitable convergency critera, singularities as well as constraints but also a higher computational effort than analytical methods. Since this thesis solves the problem by optimization, the focus will be on highlighting the

numerical rather than analytical methods due to their scalability and robustness which will be demonstrated during the next chapters and also how to overcome the previously mentioned drawbacks.

The core principle however is to iteratively compute a joint variable vector Θ weighted by μ that moves the previous configuration $\theta \mapsto \theta'$ in a way that the result obtained by forward kinematics progressively matches the Cartesian target $\mathcal{Y}$. This can be denoted by (2.12) where d is an arbitrary metric distance function under a given Cartesian objective.

$$\theta' = \theta + \mu\Theta \qquad d(f(\theta'), \mathcal{Y}) < d(f(\theta), \mathcal{Y}) \tag{2.12}$$

It is important to mention that different objectives require different metrics and which are essential for optimization techniques. Let the objective be described solely by position, so the translational distance d_t can be calculated by computing the Euclidean distance between two points p_1, p_2 defined as (2.13).

$$d_t = ||p_1 - p_2|| = \sqrt{(p_{1_x} - p_{2_x})^2 + (p_{1_y} - p_{2_y})^2 + (p_{1_z} - p_{2_z})^2} \tag{2.13}$$

Regarding the orientation objective, the computation of rotational distances depends on the mathematical representation which can be given by Euler angles or a quaternion where the former can easily be converted into the latter and vice versa. Considering a quaternion representation, the rotational distance d_r between two quaternions q_1, q_2 can be obtained by (2.14) using their dot product.

$$d_r = q_1 \cdot q_2 = 2 \arccos\left(\frac{q_{1_x} q_{2_x} + q_{1_y} q_{2_y} + q_{1_z} q_{2_z} + q_{1_w} q_{2_w}}{|q_1||q_2|}\right) \tag{2.14}$$

Considering a full pose objective including both position and orientation, numerical methods for inverse kinematics run into a multi-objective optimization problem where the performance heavily depends on the resulting Pareto front. Since rotational distances are limited but translational distances can be arbitrarily large, it is necessary to calculate a rebalanced translational distance $\hat{d}_t$ that can be achieved by (2.15). In this, l denotes the fixed length of the kinematic chain and Δ is the variable Euclidean distance from the base to the end effector depending on the current posture of the kinematic model.

$$\hat{d}_t = \frac{\pi d_t}{\sqrt{l\Delta}} \tag{2.15}$$

It is then possible to define a combined multi-objective distance d_p (2.16) that can be used for optimization approaches that require a single objective function where each included objective can further be modified by an individual weight.

$$d_p = w_t \hat{d}_t + w_r d_r \tag{2.16}$$

Many approaches to iteratively optimize the joint variable configuration compute a gradient by applying a small change $\sigma \approx 0$ to each joint variable in order to estimate the first-order derivatives with respect to the change of the end effector. This yields the Jacobian $J(\theta)$ that is a matrix that has dimensionality defined by the number of joint variables and the objective and can be calculated using (2.17).

$$J(\theta) = \begin{pmatrix} \partial\mathcal{X} \\ \partial\theta \end{pmatrix} \tag{2.17}$$

A mutable inverse of the Jacobian and the error vector $\vec{e}$ of the end effector to the target lastly give rise to the joint variable change (2.18) that must be applied to (2.12) where $\mu > 0$. Note that the error can both consist of translational and rotational information where each dimension must be treated independently.

$$\Theta = J^{-1}(\theta)\vec{e} \tag{2.18}$$

This method results in a greedy behaviour and supports fast convergency to local extrema but might get stuck in those especially for a higher degree of freedom. Other strategies try to solve the problem by means of biologically-inspired concepts where information to the problem is not obtained by the gradient but by a specifically designed suitable objective function. Those provide a impressively higher flexibility and robustness with increasingly articulated kinematic models but by exchange of higher computational effort and slower convergence.

This section was mainly responsible for introducing the fundamental definitions and mathematical foundations for inverse kinematics with particular focus on the numerical concepts. A deeper insight into the individual approaches will be given in section 2.3 as well as in chapter 3 which presents the current state of the art.

2.1.4 Degree of Freedom

In the domain of robotics, the term *Degree of Freedom* (DoF) is ambiguous and shares two different meanings. First, it describes the translational (*Surge, Sway, Heave*) and rotational (*Roll, Pitch, Yaw*) dimensions in which a particular object can move or operate. Hence, these dimensions give an upper bound by a six-dimensional degree of freedom in Cartesian space where *Surge/Roll* is related to the X-axis, *Sway/Pitch* to the Y-axis and lastly *Heave/Yaw* to the Z-axis with respect to the object's coordinate system. In this interpretation, the number of joints does not generally give rise to a robot's inherent degree of freedom since there might be redundant axes of motion that do not contribute to an overall higher degree of freedom. Second, the degree of freedom is defined by the sum of the robot's independently moveable bodies and their each individually defined motion axes resulting in distinct joint space dimensions. Accordingly, the robot's calculated degree of freedom can grow arbitarily high where a suggestion of different

methods on how to compute the resulting degree of freedom is presented in [64]. For inverse kinematics, the latter representation is mostly used in order to indicate the geometric complexity since a higher degree of freedom usually leads to an increasingly non-convex optimization problem and thus to a higher occurance of local extrema as well as singular configurations. [56, 64]

2.1.5 Joint Types and Constraints

There are many different joint types to define the motion of an articulated robot. Each joint possesses its own degree of freedom for which the specific dimensions can also be contrained. This section only introduces the most essential joint types. A more detailed overview from a larger variety that depends on the kinematic and geometric purposes is given in [64].

- *Fixed* joints allow no motion by any axis and thus have 0-DoF. Although they do not represent a real joint in the mechanical sense, they can be used for defining constant hierarchical connections between links.
- *Prismatic* joints denote 1-DoF translations along the defined axis of motion.
- *Revolute* joints denote 1-DoF rotations about the defined axis of motion.
- *Planar* joints allow 2-DoF translation and 1-DoF rotation within a plane that is perpendicular to the defined axis of motion.
- *Spheric* joints allow any 3-DoF rotation within a sphere.

All motion for a joint can be constrained which means that independent lower or upper limits for each degree of freedom can be set. This is not only greatly beneficial in order to describe realistic motion but also to restrict the dimensional domains – especially for translational motion – in the resulting joint space.

2.1.6 Singularities

The first derivative of the kinematics equations (2.1) yields the velocity and thus allows to describe the Jacobian matrix. A kinematic singularity is then observed as the Jacobian becomes rank deficient – and thus singular – which causes an instantaneous loss in the degree of freedom of the kinematic model. Basically, this means that different joint variables within the kinematic joint space do not remain independent anymore and the inverse of the matrix becomes numerically unstable.

As already mentioned in section 2.1.3, many numerical approaches to solve the inverse kinematics problem approximate the end effector gradient to the Cartesian objective by applying small changes to all joint variables. In the sense of forward kinematics, these changes would describe internal joint space singularities which are configurations that do not cause a change of the end effector in position or orientation and thus make joint motions redundant. In terms of inverse kinematics, these changes cause the calculated joint velocities diverge to infinity. External

singularities typically occur at the domain boundaries of the joint space dimensions. Those can be observed either by completely extended manipulators or by Cartesian targets that would require a configuration that is not contained within the joint space. [21, 46]

Usually, the occurence of singular configurations rises with the degree of freedom and the geometric complexity of the kinematic model. Approaches that rely on computing the Jacobian are then not able to reliably obtain solutions anymore since the gradient can not be inverted. This is where biologically-inspred iterative methods greatly benefit by solving the problem without directly incorporating inherent knowledge of them problem.

2.2 Algorithmic Methodology

There are two fundamental concepts for solving the inverse kinematics problem which have been studied for decades and were mentioned several times during this chapter. Those can be classified by their algorithmic methodology that is either analytical or numerical. This thesis follows the latter and hence will more focus on introducing the numerical strategies for which further the principle of randomization can offer great opportunities in efficiency and adaptivity.

2.2.1 Analytical

The winning fact of analytical methods is that those are rapidly fast and exact. Since the computational solution is directly derived from the structure of the kinematic model, inverse kinematics can be solved within one step and can give identical results repeatedly. Analytical methods can primarily be subdivided into algebraic and geometric approaches. The former directly rely on the transformation sequences from the base to the end effector and apply algebraic mathematical conversions in order to obtain a solution for each joint variable. The latter also derive a solution for each joint variable, but instead incorporate direct information about the spatial geometry of the manipulator which is described by the individual link lengths to their joints and can then be formulated as equations consisting of trigonometric terms. Examples for different robot manipulators on how to obtain their joint variables both algebraically and geometrically can be found in [46]. Deriving these solutions is typically eased by intersecting or parallel consecutive joint axes since these can induce redundancy for the end effector configurations. However, due to the dramatically increasing mathematical complexity and especially non-linearity not only by each additional degree of freedom but also caused by different joint types and constraints, such closed-form solutions are highly non-trivial – or even non-existent – and thus only available for simple kinematic models. Accordingly, analytical approaches become poorly scalable and impracticable for character animation due to arbitrary kinematic geometries and the degree of freedom is often higher in comparison to most industrial robots. [79, 46, 32]

2.2.2 Numerical

At first glance, numerical methods appear to be slower and considerably more expensive than analytical approaches and also do not guarantee to find an exact solution. Nevertheless, this methodology robustly allows to approximate to a solution that is still highly accurate and hence mostly remains with a negligibly small error. More particularly, it allows to find solutions which could not be obtained by analytical methods due to the complexity of the problem which has encouraged their use to solve inverse kinematics problems. Numerical analysis and optimization is a mathematical technique that has consistently proven to perform successful for various different and highly complex problems [50, 92, 41]. Those either compute functional derivatives and integrate inherent knowledge of the problem in order to provide a stable and efficient convergency or try to satisfy a predefined objective function by finding local extrema. Their most common type for inverse kinematics is given by iterative methods such as the widely used and well known Jacobian approaches as well as the Cyclic Coordinate Descent. While they are comparatively fast and more predictable, the lack of these in terms of inverse kinematics is that they follow a certain presumed model that is iteratively computing the gradient. This usually results in a far more present exploitation than exploration and thus in frequently getting stuck in suboptimal extrema. Biologically-inspired methods then again are based on optimization where information about the gradient is not neccessarily required but although can efficiently approximate to an accurate solution. While both techniques offer great scalability in contrast to analytical solutions, the challenge among these is in mastering the trade-off between accuracy and computational effort that is required in order to find a suitable solution. The use of numerical methods is particularly interesting for applications that do not rely on precisely accurate solutions where this imprecision might already be given by technical limitations such as sensor noise, but also in robotics simulation and virtual reality including motion capture and animated characters. [11, 57, 88, 58]

2.2.3 Randomization

Randomization itself represents a concept which has always been present in the course of human history although the first attempts of finding a mathematical formalization for probability theory are comparatively young. With rising computational capabilities, it constitutes a powerful tool to improve solving complex – in particular higher-dimensional – problems or even to find approximate solutions to NP-hard problems. In more detail, randomness can dramatically ease the algorithmic design and greatly reduce the required computational effort while it is still able to obtain surprisingly good results. The algorithmic class of randomized algorithms can primarily be subdivided into *Las-Vegas* and *Monte-Carlo* strategies. First provide an exact solution to the given problem while second return an approximate result constrained by a bounded error probability. Related to the inverse kinematics problem, the latter randomization methodology is common as an additional or even fundamental part of numerical approaches. More precisely,

randomness can be used to weigh the iterative change that is applied to the joint variables but also to perform an efficient guided random search within the solution space as characteristic for biologically-inspired evolutionary and collective systems. Another use is to randomly generate different initial configurations by which gradient descent approaches are then able to obtain distinct results. Accordingly, randomization greatly serves to realize an efficient search space exploration and thus to enhance the robustness and adaptivity of an algorithm. [37, 69, 52, 78, 8, 63]

In this thesis, randomization is an essential part of the algorithmic solution that is presented in chapter 4 and will particularly demonstrate its effectiveness applied to the problem of multi-objective optimization.

2.3 Biologically-Inspired Artificial Intelligence

Classical methods in artificial intelligence aim to design solutions that are able to reproduce cognitive capabilities and therefore provide a descriptive solution to a given problem. In contrast to this, biologically-inspired artificial intelligence directly emerges from various concepts and patterns that can be observed in natural phenomena. This usually results in the property that former solutions are good in solving problems that human beings show difficulties with while vice versa they do not succeed in tasks that latter approaches can easily solve and that living organisms seem to fulfill almost effortlessly. [29, 13, 65]

The most prominent and widely researched biologically-inspired models are constituted by artificial neural networks which are designed to imitate the capabilities among the individual cortical areas of the human brain. While they behave as function approximators and can be used both for regression analysis or classification, they have not only been successfully applied to stock market prediction and control systems but have also become an important area of research for pattern recognition in language processing as well as computer vision. [34, 49, 45]

Cellular systems have been developed by inspiration of biological cells in living organisms. Their simplest model is given by cellular automata which became popular by Conway's Game of Life [43, 44]. They are represented by a cellular grid where each cell can only influence its closest neighbouring cells under a predefined transition function and aim to let complex behaviour emerge from a set of simple rules.

A particular relevance of biologically-inspired approaches in this thesis is given by genetic algorithms as well as particle swarm optimization and is discussed in more detail during the following sections 2.3.1 and 2.3.2. While both share the same concept of letting a group of rather simple individuals solve a specific problem, they follow different strategies and dynamics for search space exploration observed in natural phenomena.

2.3.1 Genetic Algorithms

In biologically-inspired artificial intelligence, *Genetic Algorithms* (GA) constitute a subfield of evolutionary systems which have been introduced by J. F. Holland [35] although first motivations were already given earlier by famous computer scientists such as A. Turing [80] and J. v. Neumann [85] as well as later on in [70, 30]. They are driven by the theory of natural evolution and genetics that have been formed by C. Darwin and G. Mendel and follow the principle of *Survival of the Fittest* according to the introducing quote in chapter 1. In biology, there are four *Pillars of Evolution* which are given by *Population*, *Diversity*, *Heredity* and *Selection*. Clearly, there can not be any evolution without a population that consists of a group of individuals. Diversity then mentions that each of these individuals must have distinct characteristics which is basically defined by their genotype that is the genetic material of chromosomes. The meaning of heredity is that those characteristics can be transmitted from the ancestry to their offspring over many generations what is related to sexual reproduction. Selection lastly states that only few individuals are capable of reproducing – namely those that are "most responsive to change" – what is determined by their phenotype that performs differently successful under various environmental constraints and thus guides the evolutional progress. However, in contrast to natural evolution in which organisms have no specific global goal but many separate and independent local goals, genetic algorithms in artificial evolution are used to solve optimization problems in which the measure of individual success is predefined by a shared global objective. [29, 25]

Given a population by a number of individuals, genetic algorithms model evolution by the cycle of selection, recombination and mutation. Each individual is assigned a fitness value depending on the measurable quality of appearance and functionalities emerging from its encoded genes under a given fitness function. This general process is illustrated by Fig 2.3 and will be explained in more detail.

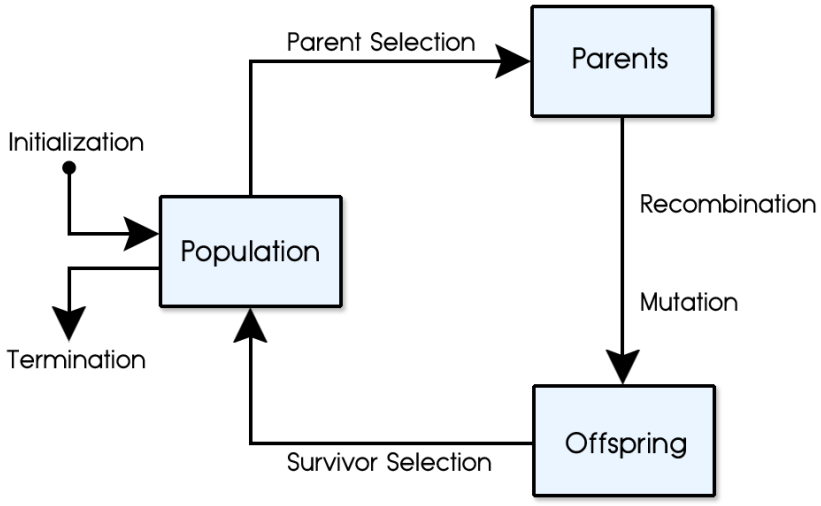


Figure 2.3: General Cycle of Genetic Algorithms

Population

The population is the set of all individuals of the current evolved generation. In this, each individual shares the same chromosome encoding scheme for the genes and represents one candidate solution with assigned fitness value under the given objective. The initialization of the population is usually random but can also be biased with purpose to advance convergence or to search for solutions only within a small neighbourhood. The algorithm terminates as soon as an individual is found whose phenotype performs well under the given problem. However, termination can also be forced by detection of the population being stuck in a suboptimal extrema or by exceeding a limited number of iterations or amount of time.

Encoding

There are various possible ways to model the encoding scheme of the genotype for the individuals and it highly depends on the given problem setting. In addition, it also affects the possible concepts for recombination and mutation that can be performed. Popular and successfully proven strategies are given by binary, discrete, real-valued, character, permutation or tree encoding. While the first four methods are quite intuitive, permutation encoding is typically applied to sorting problems and tree encoding is used to describe hierarchical structures. The representation of the genotype is then given by an array of arbitrary length that mostly defines the dimensionality of the resulting search space. In example, the real-valued encoding scheme is shown in Fig. 2.4 which is of particular importance for this thesis.

$$\Big(0.3 \mid 0.1 \mid 0.5 \mid 0.6 \mid 0.9\Big)$$

Figure 2.4: Real-Valued Encoding Scheme

Fitness

The fitness of an individual is usually calculated as a real-valued number which measures the quality of the phenotype under a predefined objective. It is required for any kind of selection during evolution but can also be used in order to estimate the diversity within the population or to define a suitable convergency criteria by assigning a threshold value.

Fitness Function

In the context of evolutionary systems, the fitness function is a synonymous use for the mathematical representation of the objective function. Although it is both possible to either minimize or maximize the fitness function, the former is usually more efficient since it is bounded to zero. Accordingly, in this thesis the characterization of fitter individuals is always associated to smaller fitness values. The design of the fitness function is usually one of the most crucial challenges since it guides the population through the progress of search space exploration and exploitation.

Parent Selection

At the beginning of one evolution cycle, the fittest individuals need to be selected to become the parents for the next generation. This imitates heredity over many evolutions in order to keep good genes within the pool of individuals. Exclusive selection of fittest individuals usually leads to premature convergency which is highly attracted to suboptimal extrema. Accordingly, a well designed parent selection frequently choses individuals with high fitness to transmit their genes but also allows weaker individuals to reproduce some of the time. Although parent selection can be performed purely at random and the evolution can still converge to same solutions, there are mainly three commonly used selection operators.

- **Roulette Wheel** – This operator is also known as fitness proportional selection. The probability of an individual to be chosen as parent is simply calculated as the ratio of its own fitness value to the sum of fitness values over the whole population denoted by (2.19).

$$p(i) = \frac{f(i)}{\sum_{i=1}^{n} f(i)} \tag{2.19}$$

  Clearly, this strategy will perform poorly if there is one individual that has a greatly higher fitness value than all others or if all fitness values are roughly equal to each other. In those cases, either the fittest individual is chosen too often or the method resembles a random selection.

- **Rank** – This operator is very similar to roulette wheel selection, but successfully avoids the mentioned drawbacks. Assuming the population is ordered by fitness values, the probability of an individual to be chosen as parent can be calculated using 2.20.

$$p(i) = \frac{n - i + 1}{\sum_{i=1}^{n} i} \tag{2.20}$$

  This method guarantees always to create a probability distribution that continuously decreases with the quality of an individual in the population and remains independent from the distribution of fitness values.

- **Tournament** – Lastly, this method randomly selects a reasonably small subset of individuals from the population and the one competing best by its fitness value is chosen as parent. All others are put back into the pool of individuals and can participate in further tournaments. This selection operator is very frequently used due to its capability of chosing individuals of high quality while remaining diversity at the same time. Hence, it does not require any prior knowledge about the underlying problem and remains dynamic within a noisy and highly non-linear solution space.

Recombination

Recombination – often also termed as crossover – simulates the process of sexual reproduction in which two parent chromosomes are randomly merged into a new offspring chromosome. It is controlled by a recombination rate parameter that determines whether this process occurs or not. This rate is usually chosen very high since it aims to find the fittest subsolution that can be constructed from the gene pool. Again, three popular recombination operators are presented in more detail.

- **Uniform** – This operator creates the new offspring chromosome by chosing each gene to be equiprobably transmitted from one of both parents. Accordingly, this typically results in an exploration within same dimensions and therefore obtains solutions within a related neighbourhood.

$$\begin{array}{c} \big(0.7 \mid \underline{1.1} \mid 0.6 \mid \underline{0.5} \mid \underline{0.8}\big) \quad \big(\underline{0.3} \mid 1.5 \mid \underline{1.4} \mid 0.9 \mid 2.2\big) \\ \downarrow \\ \big(0.3 \mid 1.1 \mid 1.4 \mid 0.5 \mid 0.8\big) \end{array}$$

Figure 2.5: Uniform Recombination

- **One Point** – This operator is very similar to uniform recombination, but instead randomly chooses a position at which the chromosome is split. The offspring chromosome is then created from the genes of the parent chromosomes that are either left or right to this position. This operator will cause more diverse solutions by exploration over different dimensions.

$$\begin{array}{c} \big(\underline{0.7 \mid 1.1} \parallel 0.6 \mid 0.5 \mid 0.8\big) \quad \big(0.3 \mid 1.5 \parallel \underline{1.4 \mid 0.9 \mid 2.2}\big) \\ \downarrow \\ \big(0.7 \mid 1.1 \mid 1.4 \mid 0.9 \mid 2.2\big) \end{array}$$

Figure 2.6: One Point Recombination

- **Arithmetic** – In some cases, it can be useful to define an arithmetic function to recombine the parent chromosomes. In this example, each offspring gene is calculated by the arithmetic mean of its parent genes. Note that this operator differs in a way that it is able to introduce new genetic material into the pool of individuals without performing mutation.

$$\begin{array}{c} \big(0.7 \mid 1.1 \mid 0.6 \mid 0.5 \mid 0.8\big) \quad \big(0.3 \mid 1.5 \mid 1.4 \mid 0.9 \mid 2.2\big) \\ \downarrow \\ \big(0.5 \mid 1.3 \mid 1.0 \mid 0.7 \mid 1.5\big) \end{array}$$

Figure 2.7: Arithmetic Recombination

Mutation

Mutation models natural errors that can occur during the recombination process of two chromosomes and applies small modificiations to the offspring genotype. In contrast to recombination, the mutation rate is calculated separately for each gene and is usually chosen very small or proportional to the dimensionality of the chromosomes in order to preserve genotypes of high quality. The design of the mutation operator should allow to escape from suboptimal extrema and to increase convergency in populated regions where recombination scores no improvements anymore.

- **Bit Flip** – This operator can only be applied to binary encoding schemes and mutates the gene by flipping its bit from one to zero or vice versa.

$$\begin{pmatrix} 1 \mid \underline{0} \mid \underline{1} \mid 1 \mid 0 \end{pmatrix} \\ \downarrow \\ \begin{pmatrix} 1 \mid 1 \mid 0 \mid 1 \mid 0 \end{pmatrix}$$

Figure 2.8: Bit Flip Mutation

- **Add Value** – This operator mutates a real-valued gene by adding a small value that is sampled from a uniform or Gaussian distribution. While the former will cause a more dynamic and explorative mutation, the latter typically results in an exploitation which still allows larger jumps within the solution space. In case of limited domain boundaries, the mutated genes must be clipped in order to prevent invalid solutions.

$$\begin{pmatrix} \underline{0.5} \mid 1.3 \mid 0.2 \mid \underline{0.7} \mid 1.1 \end{pmatrix} \\ \downarrow \\ \begin{pmatrix} 0.3 \mid 1.3 \mid 0.2 \mid 0.8 \mid 1.1 \end{pmatrix}$$

Figure 2.9: Add Value Mutation

- **Swap Content** – While mutation usually operates on a specific gene, this mutation operator performs mutation by interchanging two genes without affecting their particular values. This technique can be useful for chromosomes whose genetic encoding represents sequential information.

$$\begin{pmatrix} \underline{0.5} \mid 1.3 \mid 0.2 \mid \underline{0.7} \mid 1.1 \end{pmatrix} \\ \downarrow \\ \begin{pmatrix} 0.7 \mid 1.3 \mid 0.2 \mid 0.5 \mid 1.1 \end{pmatrix}$$

Figure 2.10: Swap Content Mutation

Survivor Selection

After recombination and mutation, the survivors must be selected which then constitute the population for the next generation. More particularly, the survivor selection reduces ψ ancestors and λ offspring to a new population of ψ individuals where the replacement strategy can crucially affect the evolvability of the population, meaning the number of generations until an optimal solution is found. Popular strategies rely on the fitness or age of the individuals and are usually extended by the concept of elitism.

- **Fitness** – This strategy simply selects the ψ fittest individuals among all ancestors and offspring as survivors for the next generation. This ensures to quickly find good solutions since the fittest individuals are never discarded but therefore lacks in exploration and can lead to premature convergency.
- **Age** – The age of an individual is given by the number of generations it has survived. Although there exist various methods to perform a replacement based on age, the most common method is to produce $\lambda = \psi$ offspring where all of them are selected as survivors and all parents are discarded. Accordingly, each individual exists for exactly one generation. This supports a highly dynamic search space exploration but can lose good solutions that have already been discovered.
- **Elitism** – The concept of elitism implies that a small number $\kappa < \psi$ of the best solutions from the previous population survive as direct copies without any changes in their genotype. The remaining $\psi - \kappa$ individuals are then selected by age or fitness. Incorporating elitism is usually recommended especially in noisy search spaces since it allows to keep the fittest solutions while the remaining individuals can serve a dynamic evolution.

Niching

In optimization problems, the occurence of multiple local extrema is a common problem and it often remains difficult to efficiently escape from suboptimal regions. Although genetic algorithms can be designed to perform impressively resilent to premature convergency, a population can also accumulate and stop gaining progress which is typically observed by a low genetic diversity or all fitness values approximating zero. Niching denotes a technique that aims to prevent such situations by forcing the population to settle in different regions simultaneously and is particularly interesting for multi-objective problems. A popular strategy to achieve such evolutional behaviour is to perform *Preselection* which removes parents that are worse than their offspring from the genetic pool. Further methods use *Fitness Sharing* within densely populated regions that are discovered by Euclidean distance thresholds. As a result, individuals within those areas will become less likely to be selected as parents. Lastly, inducing a *Crowding Distance* into the population can ensure a uniform spread of the fittest individuals along the Pareto front and therefore allow to fall into better extrema.

Designing a genetic algorithm implies no strict rules and offers many possibilities and decisions to be made in order to create an efficient optimization technique. Depending on the given problem, a suitable encoding scheme must be chosen whereby appropriate genetic operators for recombination and mutation are neccessary. Obviously, this typically requires many parameters to be specified for which no universally optimal configuration settings are available. While a recombination rate that is set to one and mutation rate of zero results in a population that will consist of multiple copies of same individuals, a recombination rate of zero and mutation rate set to one yields a persistently scattering group of individuals that will never converge to a common solution. Although the recombination rate should usually be chosen high while the mutation rate is suggested to remain extremely low, the whole evolutional strategy can be adjusted and designed to perform successfully even under highly non-linear and noisy search spaces. A great benefit of genetic algorithms to gradient descent approaches is their ability to perform a guided search for multiple solutions at the same time. This parallel search also allows to serve multi-objective optimization problems by an efficient exploration of the resulting Pareto front and to scale well for greatly higher-dimensional search spaces.

According to the *No Free Lunch* theorem [94], there is no single approach that performs best under all kind of problems what is related to the fundamental concepts of overfitting and underfitting. The former is caused by a problem tailored method that is highly biased and performs perfectly well for what it is supposed to solve but breaks as soon as anything changes. The latter yields the opposite by performing convergently equal under all given problems what lastly resembles an unbiased random search for an optimal solution. In this context, Fig. 2.11 demonstrates the flexibility of evolutionary algorithms which achieve a proper balancing between both and thus accomplish a decent algorithmic performance measure over the infinite range of problems.

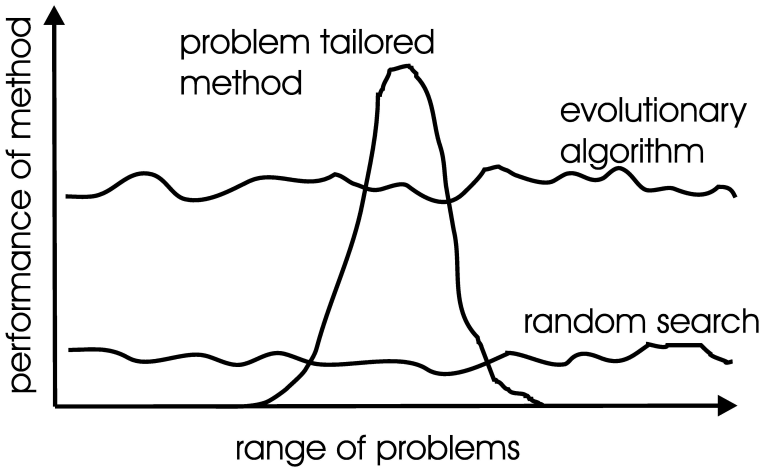


Figure 2.11: Algorithmic Performance of Evolutionary Algorithms [25]

2.3.2 Particle Swarm Optimization

Particle Swarm Optimization (PSO) constitutes an optimization technique that was developed by J. Kennedy and R. Eberhart [38] and is inspired by the social behavior of bird flocks and schools of fish. It follows the idea to let complex behaviour emerge from a group of simple organisms that collectively solve a specific problem. The organisms use aggregation that is a form of natural self-organization which can be observed by the process of gathering food where each organism knows its own success and tells it to its closest neighbours. Those subsequently can integrate this available information from all their neighbours and react by turning closer into the direction where the highest concentration of successfully performing organisms – and thus also food – can be found. This procedure procreates until all organisms settle within suitable regions of optimal local extrema. [38, 29]

In particle swarm optimization, a number of organisms – called particles – form a swarm and perform a search space exploration that can be described as a continual dynamic motion. Within each optimization step, all particles compute their magnitude and direction of motion according to a collective behaviour pattern that typically depends on the performance measure of the other particles and are updated by moving a small step along this gradient as outlined by Fig. 2.12.

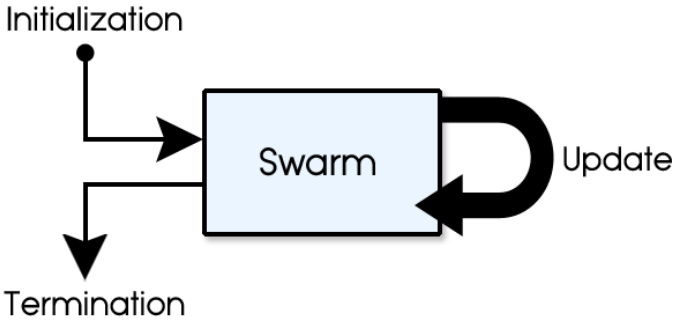


Figure 2.12: General Cycle of Particle Swarm Optimization

Swarm

Each particle within its swarm is mainly characterized by an individual position and performance value and represents one candidate solution. An additional velocity value is iteratively updated with respect to the motion update pattern. Equivalent to genetic algorithms, the particles are usually initialized by random positions in order to cover a large field of possible solutions. Also, the termination condition is satisfied as soon as a suitable solution is found or by exceeding a predefined maximum number of iterations or amount of time.

Encoding

In contrast to genetic algorithms, the encoding scheme for particles is usually given by real-valued numbers. This is due to the velocity update that is mainly designed to be applied to continuous search spaces. The candidate solutions then again are represented by arrays of arbitrary dimensionality as depicted in Fig. 2.4.

Performance

Similar to the fitness value in genetic algorithms 2.3.1, the performance value denotes the quality of the candidate solution under a given objective function.

Performance Function

The performance function again is synonymous to the objective function and also implies the same as for genetic algorithms 2.3.1.

Update

There are various methods to perform the search space exploration in particle swarm optimization. Since all strategies indicate different values to be stored within each particle or the definition of arbitrary neighbourhood functions as well as collective topologies, this section only presents the most common update type which exclusively incorporates the highest performing solution of the local history of each particle as well as the global solution that is nearest to the target.

For each particle, a personal best solution is stored in order to remember the position under which it has been most successful during exploration. In addition, the whole swarm has information about its global best solution that represents the particle that has minimum distance to the target and thus scores the highest performance among all. Accordingly, each particle always knows about its own current position x and velocity v as well as the best position x_p it has yet discovered and can tell which particle x_g within the swarm is performing most successful. The velocity update is then calculated by a simple weighted addition of these three components denoted by (2.21). This yields the new velocity v' of the particle that can be interpreted as the gradient of motion which has high probability to find an optimal solution within the underlying search space. The weighting is used to control the overall motion and behaviour of the swarm, meaning how much each particle should remain on its own velocity defined by w_1, stick to its own best discovered position controlled by w_2 and lastly how much it should be attracted by the most successfully performing particle depending on w_3. Also, an additional use of uniformly distributed random values r_p and r_g within the range $[0.0, 1.0]$ is usually suggested with intent to induce natural variations in motion and to prevent overshooting local extrema.

$$v' = w_1 v + r_p w_2 (x_p - x) + r_g w_3 (x_g - x) \tag{2.21}$$

Note that depending on the selection of the weight parameters, the velocity can grow arbitrarily high, wherefore this term is usually bounded to an upper limit. Fig. 2.13 illustrates the gradient of motion calculation for the velocity update phase.

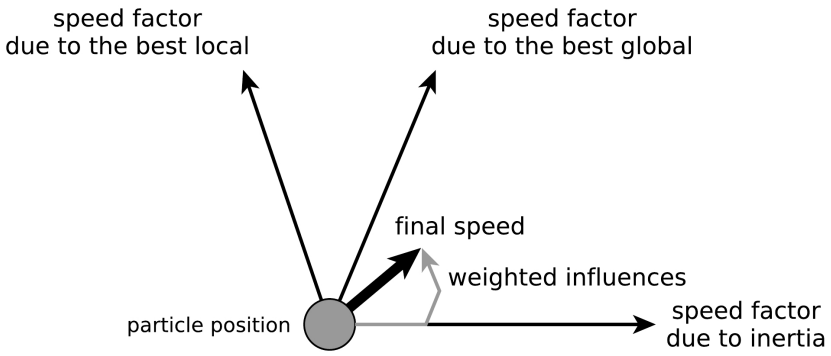


Figure 2.13: Gradient of Motion for Particles [89]

Ultimately, the updated position x' for a particle is simply obtained by adding the computed velocity in (2.21) to its previous position as denoted by (2.22).

$$x' = x + v' \tag{2.22}$$

While genetic algorithms must pass multiple stages within one iteration cycle which can all be modified differently from each other, particle swarm optimization only requires a single phase to update the candidate solutions. As a consequence, less parameters are usually required which greatly eases the algorithmic optimization. Although both concepts share many similarities and can be used to simultaneously search for multiple local extrema, the main distinction is that optimization in genetic algorithms is primarily controlled by a competition of the fittest individuals while particle swarm optimization finds a solution by a collaboration among all particles. Equivalently, the search does not require the computation of gradients and scales well with growing search space dimensionality. Popular problem-specific improvements define maximum neighbourhood distance thresholds or use social rather than geographical neighbourhoods which model subgroups of particles where communication is only allowed within same groups.

Genetic algorithms and particle swarm optimization have been successfully applied to various problems ranging from applications in control engineering and neural network training to behaviour learning in robotics but also to improvements of artificial intelligence in video games. While they are typically treated independently from each other despite of their similarities, this thesis will show that a combination can obtain much better results without requiring notably higher cost.

Chapter 3

State of the Art

"Anyone who has never made a mistake has never tried anything new."
– Albert Einstein

This chapter presents the current state of the art in solving inverse kinematics with a focus on the numerical approaches. Each method is presented separately including various related experimental results in order to show typical performance measures.

First, section 3.1 introduces popular strategies that rely on calculating the Jacobian which is involved by several different submethods to solve the inverse kinematics problem.

Section 3.2 then describes the Cyclic Coordinate Descent which is a very similar and much simpler method than the Jacobian.

Subsequently, section 3.3 outlines the comparatively novel FABRIK algorithm which is particularly interesting in terms of realistic motion.

Section 3.4 gives an overview of several results obtained by artificial neural networks.

Lastly, sections 3.5 and 3.6 presents related approaches for solving inverse kinematics based on genetic algorithms and particle swarm optimization.

3.1 Jacobian Solvers

Calculating the Jacobian (2.17) outlines the most popular and traditional strategy to numerically solve the inverse kinematics problem. As described in section 2.1.3, the inverse of the Jacobian multiplied with the error from the end effector to the target (2.18) with respect to a Cartesian objective then gives rise to the joint variable change (2.11). However, calculating the inverse is not trivial since the matrix is – depending on the degree of freedom – usually not invertible. In particular, a pure inverse of the Jacobian can only be derived if the number of joint variables equals the dimensionality of the error. Accordingly, many manipulators are intentionally designed with exactly six degrees of freedom to provide a pure inverse to a full pose error containing position and orientation. Nevertheless, alternative solutions are available which are given by simply computing the conjugate transpose J^T [5, 93] or the *Moore-Penrose* pseudoinverse J^+ [91, 57] that is defined as (3.1).

$$J^+ = J^T(JJ^T)^{-1} \tag{3.1}$$

This achieves that the Jacobian becomes invertible in most cases except for configurations that are within a close neighbourhood to singularities and which cause the matrix to become unstable. In order to avoid such problems, the damped least squares (DLS) – also known as *Levenberg-Marquardt* – method is commonly used and was firstly applied to inverse kinematics in [87, 33]. Instead of computing the joint variable change Θ directly, it aims to minimize the quantity (3.2) with λ as a non-zero damping constant.

$$||J\Theta - \vec{e}||^2 + \lambda^2||\Theta||^2 \tag{3.2}$$

This allows to provide a numerically stable method to calculate Θ but requires a proper selection of λ for which good suggestions are given in [12, 33, 55, 14]. In addition, the computation of the pseudoinverse can be further improved by using singular value decomposition (SVD) which then allows to design a selectively damped least squares method as presented in [10]. The pseudoinverse of the Jacobian can then be calculated using (3.3) which allows to obtain similar performance as for the damped least squares method and does not require a damping constant to be chosen.

$$J^+ = VD^+U^T \tag{3.3}$$

However, all Jacobian methods share the same downside of running into suboptimal extrema which is caused by the nature of gradient descent strategies. To overcome this issue, there exist various suggestions on performing frequent restarts from random initial configurations. Recently, in [8] a novel method called TRAC-IK was presented that is able to obtain promising results and is publicly available under the ROS framework [31, 7]. It was compared to the pseudoinverse Orocos KDL

method [40] concerning success rate and average computation time with respect to the degree of freedom of the manipulator. Selected results are listed in Tbl. 3.1 and were obtained from 10.000 randomly generated and reachable joint configurations with respect to a pose objective at 10^{-5} Cartesian error. From a statistical point of view, TRAC-IK strikingly outperforms the Orocos KDL inverse kinematics solver both in terms of success rate as well as computation time. However, it is not clear how many restarts are carried out which then again might result in disproportionally high joint displacements even for small changes within the target.

Manipulator	DoF	Orocos KDL	TRAC-IK
Baxter Arm	7	61.07% (2.21ms)	99.17% (0.60ms)
Jaco2	6	26.23% (3.79ms)	99.51% (0.58ms)
KUKA LBR iiwa 14 R820	7	37.71% (3.37ms)	99.63% (0.56ms)
PR2 Arm	7	83.14% (1.37ms)	99.84% (0.59ms)
UR5	6	35.88% (3.30ms)	99.55% (0.42ms)

Table 3.1: Orocos KDL and TRAC-IK Performance on Robot Manipulators [7]

Further, [75] investigated the performance of the transpose and damped least squares methods in the application of real-time motion capture. The algorithms were tested both on an unconstrained as well as constrained human skeleton model with the major goal to simulate realistic motion. In this, the final kinematic posture implicitly results from solving the inverse kinematics problem independently for each bone (1-3 DoF) under a Cartesian position objective. Still, errors of approximately 20cm in average but also up to 47cm were reported and the computation time noticeably increased when including constraints according to Tbl. 3.2.

Method	Constraints	Computation Time	Elbow Error
Transpose	Unconstrained	[0.17ms – 2.59ms] ∅0.60ms	[2cm – 47cm] ∅27cm
DLS	Unconstrained	[0.17ms – 4.16ms] ∅0.54ms	[1cm – 46cm] ∅24cm
Transpose	Constrained	[0.17ms – 23.21ms] ∅3.69ms	[6cm – 31cm] ∅20cm
DLS	Constrained	[0.17ms – 24.69ms] ∅3.48ms	[6cm – 41cm] ∅20cm

Table 3.2: Jacobian Transpose and Damped Least Squares Performance for Real-Time Motion Capture [75]

The performance among the Jacobian transpose, DLS and also SVD-DLS methods was examined in [3] when applied to an unconstrained 10-DoF kinematic chain. Both the averagely required time with respect to a given convergence distance threshold for the end effector as well as the optimization progress to the number of iterations were considered. The results are depicted in Fig. 3.1 and show that the transpose method is clearly outperformed by both other strategies. However, it is noticeable that the computation time measurements were much higher than those obtained in [8] and are extremely divergent especially for increasingly accurate results for which the transpose method performs very poorly.

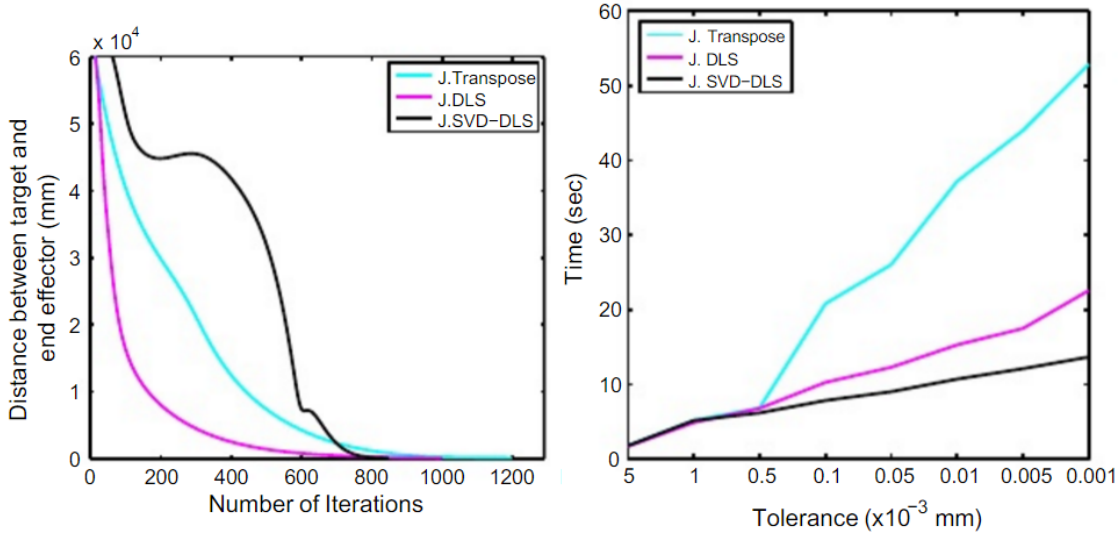


Figure 3.1: Jacobian Transpose, DLS and SVD-DLS Performance on a 10-DoF Kinematic Chain [3]

3.2 Cyclic Coordinate Descent

Cyclic Coordinate Descent (CCD) denotes an iterative heuristic optimization technique that was firstly applied to the inverse kinematics problem in [88]. Since then, it has gained much acknowledgment and became increasingly popular due to its simplicity as well as computational efficiency and is not only frequently used in robotics but also in the computer graphics and video game industry by animating highly articulated models [39, 58, 75, 48]. In contrast to Jacobian methods, the joint variables are adjusted successively for each dimension and do not require a matrix to be inverted. Instead, the optimization is done geometrically and is numerically stable. While iterating from the end effector to the base, each joint variable is repeatedly updated until the algorithm has converged to a suitable configuration as depicted in Fig. 3.2. Within each iteration, the offset between the vectors from the current joint to the end effector as well as to the Cartesian target give rise to the gradient direction for the joint variable dimension and can heuristically be weighted by an incremental strength in order to prevent oscillations.

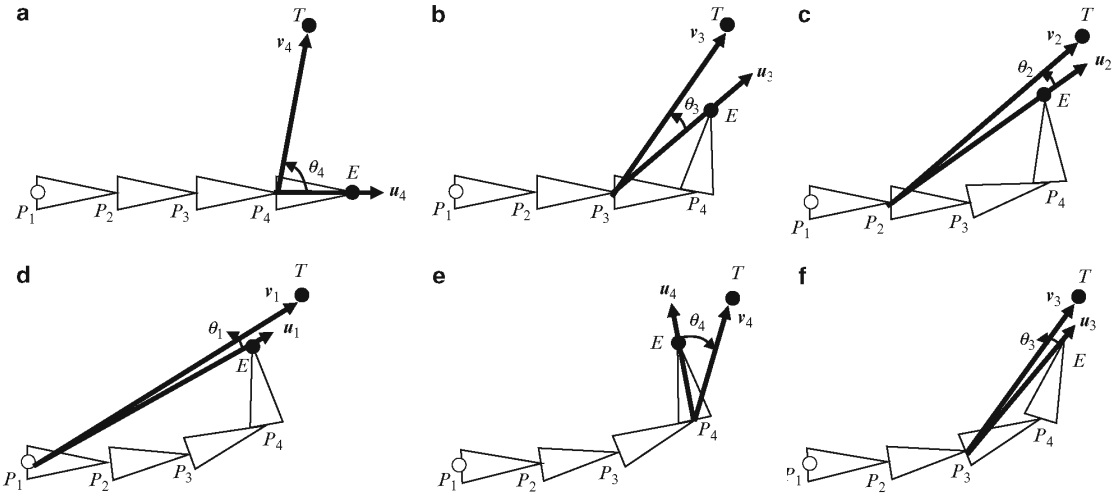


Figure 3.2: Visual Example for the Cyclic Coordinate Descent Algorithm [59]

However, it is often criticised that this approach tends to produce unrealistic motion independently from incorporating joint constraints which is mainly caused by overemphasizing joint variables that are geometrically closer to the end effector [3, 39, 48]. Also, it shares the same problem of running into suboptimal extrema and is specifically designed to operate on serial chains.

In [60], a performance study regarding the success rate of the algorithm on a 5-DoF kinematic chain was performed where more than 90% of randomly generated joint configurations could successfully be approximated. Further, experiments to those in [75] have also been executed for the Cyclic Coordinate Descent method for which Tbl. 3.3 extends the results shown in Tbl. 3.2. It was possible to achieve better results in terms of computation time and slightly smaller errors in average could be obtained when using constraints although the error spikes increased from 46cm to 54cm (unconstrained) and 31cm to 47cm (constrained) respectively.

Method	Constraints	Computation Time	Elbow Error
CCD	Unconstrained	[0.16ms – 1.07ms] ∅0.41ms	[1cm – 54cm] ∅32cm
CCD	Constrained	[0.16ms – 5.08ms] ∅1.40ms	[4cm – 47cm] ∅18cm

Table 3.3: CCD Performance for Real-Time Motion Capture [75]

Further measurements in [3] on an unconstrained 10-DoF kinematic chain confirm that the CCD algorithm is clearly able to outperform the Jacobian transpose, DLS as well as SVD-DLS methods. Instead of requiring multiple seconds to achieve precise accuracy as shown in Fig. 3.1, computation times of 123ms with 4.69ms per iteration were possible as indicated in Fig. 3.4 allowing to support interactive frame rates.

3.3 FABRIK

In contrast to Jacobian approaches 3.1 and CCD 3.2, FABRIK (Forward And Backward Reaching Inverse Kinematics) [3] outlines a novel approach that solves the inverse kinematics problem by iteratively applying geometric heuristics. It has been successfully used in virtual character animation as well as analytical 3D motion analysis and motion tracking and thus has proven to constitute a valuable solution in computer graphics simulation [58, 2, 27, 4]. The joint positions are obtained by locating points on lines while first iterating along the forward and then along the backward direction of the kinematic chain. Each joint is adjusted at one time as illustrated in Fig. 3.3. While the forward iteration tends to pull the root away from its position, the backward iteration compensates this behaviour by treating the root as the new end effector.

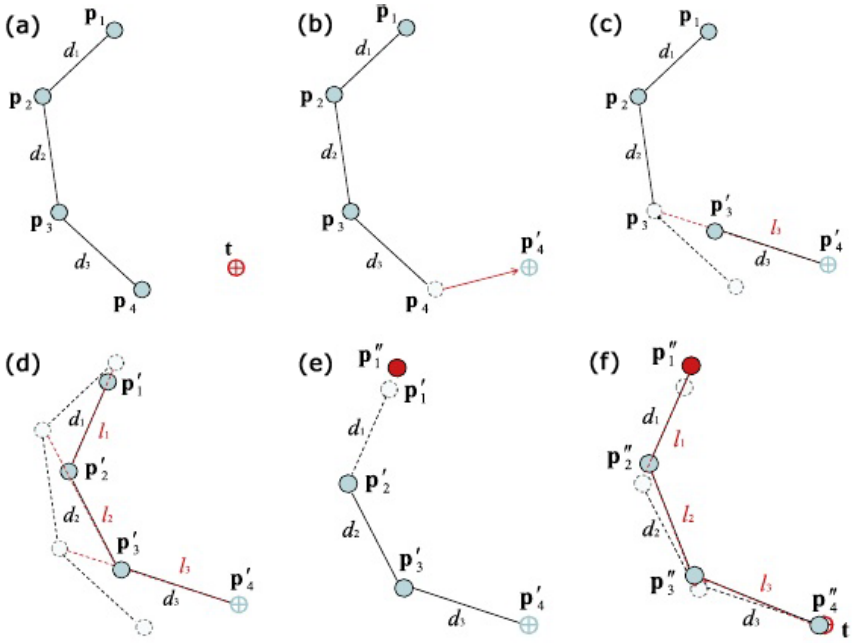


Figure 3.3: Visual Example for the FABRIK Algorithm [3]

Lastly, this allows to simulate elastic movements which therefore usually result in more realistic animations than those obtained by Jacobian or CCD approaches. A major benefit of this solution is given by its ability to handle multiple end effectors simultaneously. However, since the problem is solved exclusively in position space, problems have been observed when incorporating joint orientation constraints and it was difficult to satisfy both position and orientation of the end effector [58] although suggestions in order to improve stability were given in [3, 2]. Also, several modifications must be done when using this algorithm with prismatic joints [3] which have not been supported before in [2], which makes this solution less generic.

Experiments in [3, 2] have particularly shown that FABRIK is able to obtain strikingly better results than CCD as well as all presented Jacobian methods. Fig.

3.4 extends the results depicted in Fig. 3.1 by which FABRIK seems to show no efforts in handling increasingly desired accuracy and is able to converge within fewer iterations than CCD. However, [58] mention that the computation time per iteration becomes higher and especially when joint constraints are incorporated.

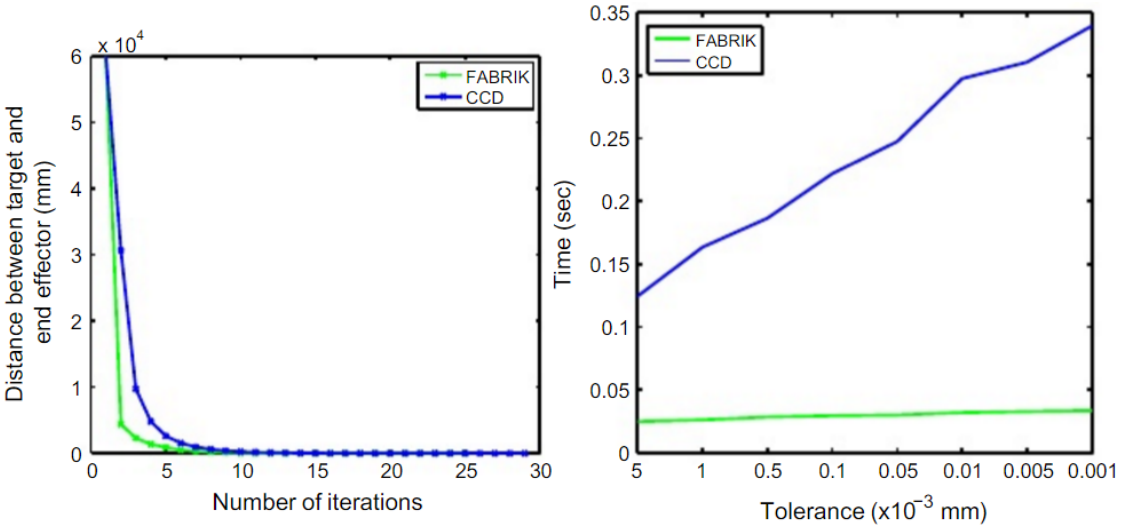


Figure 3.4: FABRIK and CCD Performance on a 10-DoF Kinematic Chain [3]

Fig. 3.5 demonstrates a low rate body tracking using FABRIK where the upper images (a) show the true and the lower images (b) the estimated body postures for which average errors of 5.9cm were reported.

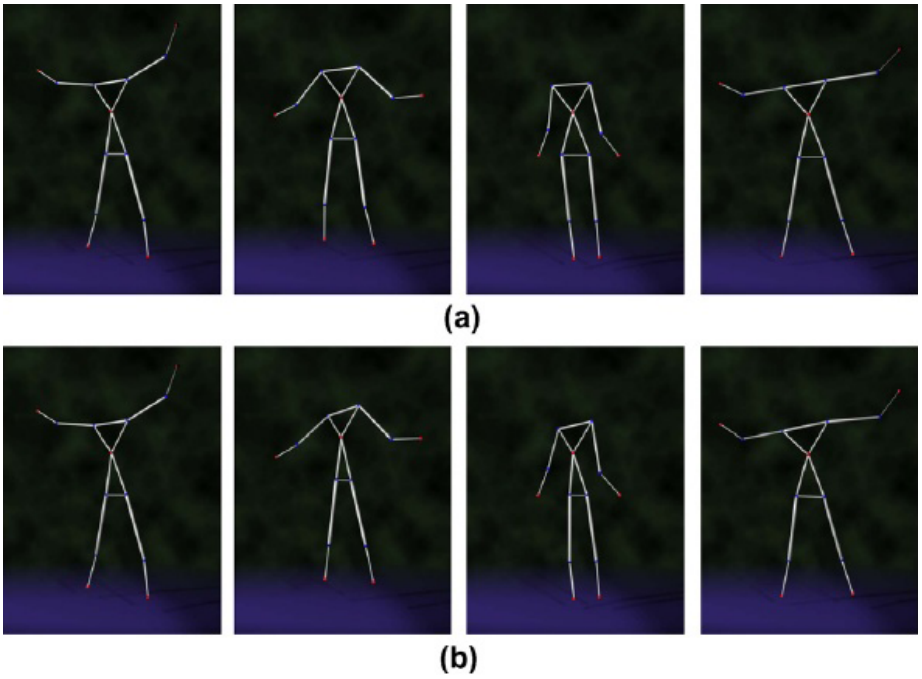


Figure 3.5: FABRIK Performance in Body Tracking [3]

3.4 Artificial Neural Networks

While analytical methods for inverse kinematics are computationally fast and exact but not generally available and numerical methods imply the opposite, artificial neural network (ANN) approaches can be classified as somewhere in between. Although there is comparatively few utilization in contrast to those methods presented in the previous sections, multi-layer pereceptron (MLP) architectures have been used to learn rather than calculate the inverse kinematics function for various robots [16, 36, 28, 42]. Once this function is obtained, a fast and straightforward computation of the joint variable configuration can be performed. Given a set of training data, a MLP can approximate the underlying function by learning the weights between the interconnected neurons of each layer using the backpropagation algorithm. However, a major problem is given by designing a proper training phase which can result in a very slow process especially for increasingly complex geometry and dimensionality. The choice of training samples remains difficult and might not be good enough to obtain sufficiently low error rates. In more detail, a low error rate during the training phase does not neccessarily give rise to a good performance when arbitrary data is used. Accordingly, those methods are typically considered unsuitable for robotic applications in which high precision rates are desired. Nevertheless, several successful strategies were presented for solving the problem of continuous trajectory planning.

In [36], a MLP was trained on the SCARA manipulator to learn the inverse kinematics function that maps from a Cartesian position to a 4-DoF joint configuration. Although only a Cartesian position objective was considered, the resulting MSE in joint configuration resolution remained at 10^{-3} in average.

Further, in [16] a neural network architecture consisting of multiple MLP subnetworks was proposed to solve the inverse kinematics problem and was tested to reach Cartesian positions for a planar 2-DoF revolute manipulator. However, comparable results as in [36] were obtained.

Lastly, in [28] the trajectory planning problem was investigated on the PUMA 560 manipulator using an extreme learning machine (ELM) which was shown not only to be impressively faster for training the network but also allowed to obtain surprisingly small errors. It was learned to map from a Cartesian input position and orientation to a 6-DoF joint configuration for which the final Cartesian error for each track point along the trajectory was about 10^{-9} in average.

Although ANN solutions can directly be used both for prismatic as well as revolute joints, the learning is mostly very limited to a comparatively low degree of freedom and difficulties are already observed even when exclusively a target position without consideration of orientation is requested. In addition, the inverse kinematics functions must be learned in advance which usually causes them to be impracticable for interactive applications.

3.5 Genetic Algorithms

Over many decades, genetic algorithms (GA) have been researched for their applicability in robotics where first relevant work adressing the inverse kinematics problem was presented in [63]. It was shown that target positions for redundant robots can robustly be matched without requiring artificial joint constraints as those are handled directly. Later, the utilization of niching methods has been investigated in order to find multiple solutions for desired end effector poses and it was possible to obtain extremely small errors of approximately zero both for position and orientation [76, 77]. Opportunities for computational efficiency were also proposed in [1] since the whole evolutionary process offers to be highly parallelized.

Lately, the capabilities of genetic algorithms for inverse kinematics and also in the application of continuous trajectory planning were extensively researched in [84]. Since related approaches often encountered difficulties with comparatively low convergency rates and long computation times, a memetic variant of genetic algorithms was presented which was shown to be able to obtain higher accuracy within less generations on serial robot manipulators and anthropomorphic robot hands. Various experiments were performed including a planar as well as the PUMA 560 and the PA10-7C robot manipulators. The measurements were recorded under full Cartesian pose targets within the reachable workspace space of the end effector. The results and are shown in Tbl. 3.4 in which GEN denotes the number of evolved generations and E_P and E_O are the position and orientation errors and t is the elapsed computation time. Also, experiments for the Shadow Dexterous Hand C6M (20-DoF) were conducted for which comparable results could be obtained regarding its higher degree of freedom. Although the required computation time to reach those highly accurate end effector poses was quite high, it was also possible to achieve reasonable solutions within far shorter time frames. In addition, a parameter trade-off was mentioned for which faster convergency rates increased the probability to get stuck in suboptimal extrema and lowered the success rate since no suitable solution could then be obtained. Alltogether, memetic GA solutions were found to be a highly flexible and generic solution to the inverse kinematics problem of robot manipulators.

Model	DoF	GEN	E_P in mm	E_O in deg	t in s
Planar	3	46.85 ± 8.03	$1.47\text{E}^{-3} \pm 6.33\text{E}^{-3}$	$3.04\text{E}^{-4} \pm 8.28\text{E}^{-4}$	0.0425
PUMA 560	6	117.55 ± 21.83	$2.18\text{E}^{-2} \pm 4.22\text{E}^{-1}$	$7.83\text{E}^{-2} \pm 9.50\text{E}^{-1}$	0.2992
PA10-7C	7	121.59 ± 54.21	$8.06\text{E}^{-2} \pm 1.33\text{E}^{0}$	$5.32\text{E}^{-3} \pm 1.41\text{E}^{-1}$	0.5493

Table 3.4: Memetic GA Performance on Planar, PUMA 560 and PA10-7C Robot Manipulators [84]

3.6 Particle Swarm Optimization

Regarding all previously introduced methods in this chapter, particle swarm optimization (PSO) is a comparatively new optimization technique for which nevertheless a lot work has yet been proposed concerning inverse kinematics. Similar benefits as for genetic algorithms were consistently reported due to their generic and adaptive optimization strategy for which no special conditions are required. In [24], PSO was applied to a 6-DoF robot manipulator under a Cartesian position objective for which very small errors within few milliseconds could be obtained. Further, in [74, 73] a swing leg motion generation for bipedal walking on a simple robot was realized and it was statistically shown to be a valuable technique for path planning and trajectory optimization under given target positions. A hybrid method combining GA and PSO was presented in [90] and found to be more efficient than the individual approaches. Another method that also successfully avoids obstacles within the environment was proposed in [72] where success rates of 100% could be achieved. Recently, the dimensional scalability of PSO for inverse kinematics of hyper-redundant manipulators – such as depicted in Fig. 3.6 – has been investigated in [15]. The performance results are listed in 3.5 and demonstrate the capabilities of PSO to achieve reasonably small error rates in average both for position E_P and orientation E_O of the end effector even under greatly higher-dimensional degree of freedom where I denotes the number of required iterations. Although the required computation time t was relatively high, many traditional methods might have failed to obtain any solution of such dimensionality at all.

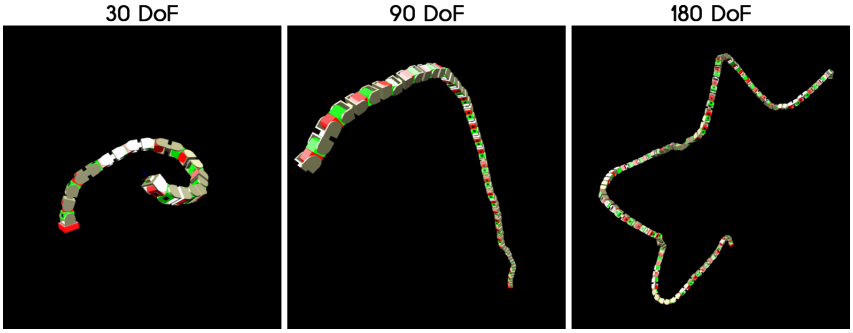


Figure 3.6: Hyper-Redundant Manipulators [15]

DoF	$\varnothing I$	$\varnothing E_P$ in m	$\varnothing E_O$ in deg	$\varnothing t$ in s
30	275.84	0.00046	0.00314	1.57
90	533.055	0.00036	0.00210	7.46
180	947.255	0.00032	0.00237	37.03

Table 3.5: PSO Performance on Hyper-Redundant Manipulators [15]

Chapter 4

Algorithmic Approach

"Many of the things that seem impossible now will become realities tomorrow."
– Walt Disney

Constructing on the fundamental knowledge provided in chapter 2 as well as on the insights into the current state of the art in chapter 3, this chapter presents the *Hybrid Genetic Swarm Algorithm* that was developed during this thesis.

First, section 4.1 gives a problem statement that highlights the particular challenges that were observed within related work and which are relevant to define specific requirements and performance measures that are to be fulfilled.

Section 4.2 continues with an overview and motivation of the complete algorithmic approach in order to ease the comprehensibility within the subsequent sections.

Section 4.3 then presents the essential aspects of this thesis. In this, the complete algorithmic design is discussed in detail and introduces several improvements and modifications to the traditional approaches which are collectively able to achieve high accuracy, fast convergence speed and significant robustness for solving the inverse kinematics problem efficiently.

Lastly, section 4.4 concludes with a short discussion on the algorithmic approach which also summarizes the remaining parameters that are important for the subsequent experimental analysis in chapter 5.

4.1 Problem Statement

Previous researches and discussions pointed out that there is no single solution to the inverse kinematic problem that performs impressively well under any performance aspects. A common goal typically is to achieve a high pose accuracy within a given short computation time. Although this can easily be obtained for kinematic models of simple geometry and lower-dimensional degree of freedom under which analytical solutions can be derived, this strategy is not generic and shows high difficulties for increasing kinematic complexity. Accordingly, numerical approaches are mainly dominating the state of the art among which Jacobian solutions as well as the Cyclic Coordinate Descent method offer great capabilities. However, those have been observed to perform less convincing for highly constrained geometry under which the occurrence of suboptimal extrema and singular configurations increases. As a consequence, several issues such as unrealistic motion or comparatively low success rates and only few dimensional scalability for higher degree of freedom have been reported. While FABRIK was able to overcome some of these problems and could obtain solutions involving multiple end effectors simultaneously, it is primarily designed for revolute joints and problems were observed in matching the orientation of the desired end effector pose. Further, artificial neural networks alltogether showed least performance in efficiently obtaining solutions on arbitrary kinematic models. This can not only be observed by several related attempts that already struggle in achieving high accuracy for an exclusive position objective but also since there are no solutions available that consider a higher-dimensional degree of freedom. In contrast to this, both genetic algorithms as well as particle swarm optimization showed no difficulties with increasing kinematic dimensionaity and different joint types for which constraints can directly be incorporated. Also, similar computation times could be achieved for comparable kinematic complexity. Nevertheless, a proper parameter selection is required and the performance heavily depends on the specific algorithmic design.

In summary, there are five decisive criteria that can be defined to measure the performance of an inverse kinematics approach.

- **Success** – A solution that is existent can also be found.
- **Accuracy** – The solution shall be as precise as required.
- **Time** – The solution shall be found as fast as possible.
- **Displacement** – The distance between consecutive solutions shall be as minimal as possible.
- **Flexibility** – The algorithm maintains high robustness, scalability and fast convergence for greatly varying kinematic and environmental requirements.

Since this thesis primarily focuses on interactive applications, a reasonably sufficient accuracy in real-time must be obtained in order to support interactive frame rates and also a high flexibility under arbitrary kinematic models is required.

4.2 Complete Overview

The aim of this thesis is to solve to problem of inverse kinematics as universal as possible. In this context, genetic algorithms constitute a very flexible as well as highly customizable optimization strategy and thus offer great opportunities to serve as the basis for the algorithmic approach. The concept of particle swarm optimization is very similar and provides collective dynamics that can easily be integrated into the evolutionary process. Such collective behaviour can be imagined in form of adoption by offspring to their ancestors which is conducted over lifetime and implicitly part of any natural evolution. The integration of both approaches results in a symbiotic and biologically plausible combination for which recombination and mutation define the genetic material and adoption allows to let individual behaviour emerge to guide the evolution. Further, a niching technique between offspring and parents is used in order to encourage exploration for multiple local extrema simultaneously and what particularly improves robustness for multi-objective optimization. An efficient heuristic exploitation is then performed on the fittest individuals among the population with intent to increase convergency and accuracy for potentially good solutions. Lastly, a heuristic wipe of the whole population can occur if all niches are detected to be stuck in dead-end paths. Intuitively, the developed algorithm combines the individual strengths of genetic algorithms and particle swarm optimization and incorporates the benefits of local search strategies. This allows to achieve a dynamic and adaptive balancing of exploration and exploitation. The single stages of the algorithmic approach are illustrated in Fig. 4.1 and will be discussed in detail within the following section. All decisions on the algorithmic design are conducted with the intent to require as few parameters as possible and to let the algorithm adaptively react to arbitrary kinematic models.

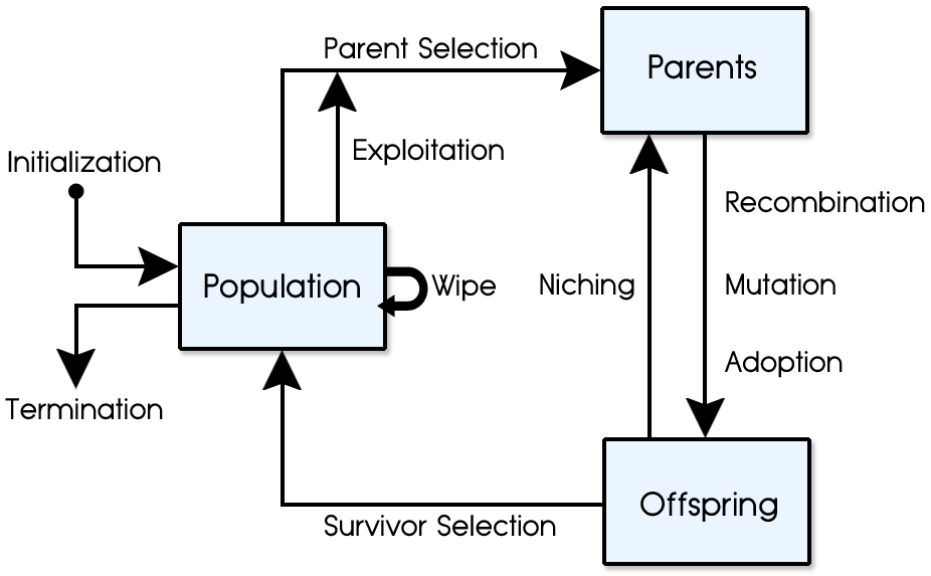


Figure 4.1: Hybrid Genetic Swarm Algorithm

4.3 Hybrid Genetic Swarm Algorithm

This section describes the algorithmic design of the *Hybrid Genetic Swarm Algorithm* (HGSA). First, the genetic encoding scheme (4.3.1) of the individuals and the fitness function (4.3.2) are introduced. Those are fundamental for the realization of the succeeding general phases in genetic algorithms covering the selection of parents (4.3.3) and survivors (4.3.8), recombination (4.3.4) and mutation (4.3.5). These also involve the phase of adoption (4.3.6) that is inspired by particle swarm optimization and the ingeration of a niching technique (4.3.7) as well as an exploitation method (4.3.9) to perform a local search. Lastly, the initialization phase (4.3.10), termination condition (4.3.11) and wipe criterion (4.3.12) are described. All successively introduced mathematical symbols and definitions are consistently used during this chapter and are summarized in Tbl. 4.1.

θ	Joint variable configuration
Φ	Evolutionary solution
$E_{\{P,O\}}$	Errors in position and orientation
$\mathcal{Y}$	Evolutionary target
Ψ	Population of size ψ with $\Psi_{1,\dots,\psi}$
$\mathcal{K}$	Elites of size κ with $\mathcal{K}_{1,\dots,\kappa}$
Γ	Mating pool of size γ with $\Gamma_{1,\dots,\gamma}$
$\mathcal{P}$	Parents of size 2 with $\mathcal{P}_{\{1,2\}}$
Λ	Offspring of size λ with $\Lambda_{1,\dots,\lambda}$
$\mathcal{X}$	Phenotype
f	Phenotype function
x	Genotype
n	Dimensionality of the Genotype
g	Evolutionary gradient
ξ	Extinction factor
π	Fitness
Ω	Fitness function
$\{\cdot\}^{\{\mathcal{P}_{\{1,2\}},*,\dagger\}}$	Property of a parent, fittest or worst individual
Ξ	Clipping function
$\mathcal{E}$	Exploitation function
$\mathcal{S}$	$\mathcal{S}_{\mathcal{P}}$: Parent selection operator $\mathcal{S}_{\mathcal{S}}$: Survivor selection operator
$\mathcal{R}$	Recombination operator $\mathcal{R}_r$: Rate
$\mathcal{M}$	Mutation operator $\mathcal{M}_r$: Rate $\quad$ $\mathcal{M}_\alpha$: Factor $\quad$ $\mathcal{M}_\beta$ Strength
$\mathcal{A}$	Adoption operator
$\mathcal{N}$	Niching operator

Table 4.1: Mathematical Symbols

4.3.1 Encoding

In the application of inverse kinematics, the encoding scheme for the individuals can be chosen relatively straightforward. Let x define the genotype of an individual, so each gene x_i can be assigned a real-valued number that is related to a corresponding joint variable at the i-th index within the kinematic chain consisting of n degrees of freedom. Hence, every genotype (4.1) implicitly represents a certain n-dimensional consecutive joint variable configuration θ as introduced in section 2.1.

$$x = \Big(x_1 \mid x_2 \mid x_3 \mid ... \mid x_{n-1} \mid x_n \Big) \tag{4.1}$$

Obviously, this encoding scheme easily allows to incorporate constraints where genes are clipped if they exceed their domain-specific limits. Those are denoted as $n_{i_{min}}$ and $n_{i_{max}}$ respectively. More particularly, the use of constraints is even advantageous since it allows to restrict the complexity of the underlying search space. Also, algebraic vector calculations can be directly performed interpreting each genotype as a n-dimensional vector in Euclidean space.

4.3.2 Fitness Function

Given a genotype x, its emerging phenotype $\mathcal{X}$ (4.2) relates to a Cartesian configuration and can be obtained by the function f which computes the forward kinematics identically to (2.7) as discussed in section 2.1.2.

$$\mathcal{X} = f(x) \tag{4.2}$$

The fitness π (4.3) of an individual with respect to the evolutionary target $\mathcal{Y}$ is then calculated by the fitness function Ω that is a function of the phenotype $\mathcal{X}$ of the individual which implicitly relates to a Cartesian configuration.

$$\pi = \Omega(\mathcal{X}) \qquad \Omega : \begin{cases} d_t(\mathcal{X}, \mathcal{Y}) & \textit{Position} \\ d_r(\mathcal{X}, \mathcal{Y}) & \textit{Orientation} \\ w\hat{d}_t(\mathcal{X}, \mathcal{Y}) + (1-w) d_r(\mathcal{X}, \mathcal{Y}) & \textit{Pose} \end{cases} \tag{4.3}$$

The fitness function for the evolutionary optimimzation is designed to converge for an end effector pose as well as exclusively for either position or orientation by being minimized. In case of position, the fitness calculation is pretty much straightforward by just computing the raw translational or rotational distances d_t (2.13) or d_r (2.14) respectively. In case of an end effector pose, the translational distance is rebalanced to $\hat{d}_t$ (2.15). Further a simple but very neat trick is applied which introduces a uniformly distributed randomized weight w between $[0.0, 1.0]$ to control the positional and rotational weights $w_p = w$ and $w_r = (1 - w)$ of the initial pose distance function (2.16).

At first glance, this strategy might give rise to doubts about its success. Solving the inverse kinematics problem for an end effector pose can be considered as a multi-objective optimization problem where first is the position and second the orientation. In those, the occurrence of local extrema dramatically increases and challenges remain in approximating the Pareto front what requires to design a suitable objective function that reliably allows to escape from suboptimal extrema in order to ensure a dynamic search space exploration. In the context of genetic algorithms, this randomization is biologically plausible and directly supports Darwin's statement to the fittest individuals, namely those that are "most responsive to change". Indeed, the fittest individual is the one who performs best regardless of the chosen weights and thus maintains a fitness value of zero computed by Ω. Accordingly, a randomized weighting of partial objectives within each fitness evaluation will cause individuals that previously performed well to be sorted out if they do not succeed when facing different criteria. Combining both discussions, randomized weights literally simulate a dynamically changing environment with varying conditions to survive as it is present in natural evolution.

4.3.3 Parent Selection

The strategy to select the parents for new offspring is chosen to depend on their rank where the intrinsic selection probabilities are calculated as denoted by (2.20). Accordingly, the parent selection operator $\mathcal{S}_\mathcal{P}$ can be formulated as (4.4) where the probabilities are normalized within the range $[0.0, 1.0]$ and alltogether sum up to 1.0. The first i accumulated ordered probability values that become greater or equal to a uniformly distributed random value between $[0.0, 1.0]$ can then determine the corresponding individual Γ_i within the mating pool Γ to be selected as parent $\mathcal{P}_{\{1,2\}}$ for the subsequent recombination phase. Note that this mating pool is initially equivalent to the population Ψ which is of particular importance for sections 4.3.7 and 4.3.8 in which the niching and survivor selection is discussed.

$$\mathcal{S}_\mathcal{P} : \mathcal{P}_{\{1,2\}} \leftarrow p(\Gamma_i) = \frac{\gamma - i + 1}{\sum_{i=1}^{\gamma} i} \tag{4.4}$$

Assuming a roulette wheel selection, this performs well if the optimization problem involves a convex search space what is typically not the case for inverse kinematics on arbitrary geometry. More particularly, the occurence of multiple local extrema is usually present. In consequence, the evolution would frequently run into a premature convergence within suboptimal regions since the fittest individual might almost exclusively be selected. A tournament selection would successfully overcome this problem but constantly result in a slower convergency and might frequently cause high joint displacements since it is less sensitive to local extrema. Also, a proper choice of the tournament size parameter would be neccessary. In contrast, a rank selection strategy serves all these issues in a good balance, scales well with arbitrary population size and remains independent to the distribution of fitness values.

4.3.4 Recombination

The genetic recombination operator $\mathcal{R}$ is designed to produce a weighted arithmetic mean of both parent genotypes $x^{\mathcal{P}_{\{1,2\}}}$ for each gene x_i that is further affected by their evolutionary gradients $g^{\mathcal{P}_{\{1,2\}}}$ and can be formulated as (4.5). In this, w_i denotes a uniformly distributed random weight within the range $[0.0, 1.0]$ that is evaluated independently for each dimension. Intuitively, first a dynamic recombination of genotypes is obtained where each gene represents a randomized interpolation between the genes of its parents. Subsequently, the evolutionary gradients are added with each independently evaluated uniformly distributed random values $r_{\mathcal{P}_{\{1,2\}}}$ that are also within the range $[0.0, 1.0]$. Those partially simulate the constant motion dynamics of particle swarm optimization. The major idea is to let offspring dive a little deeper into the direction that caused improvement within their parents, as will be discussed in more detail in sections 4.3.6 and 4.3.9. The recombination is assigned a constant rate $\mathcal{R}_r$ of 1.0. Hence, within each evolution the whole population must completely reproduce itself what has been particularly chosen with respect to the mutation phase that is described in section 4.3.5.

$$\mathcal{R} : x_i = w_i x_i^{\mathcal{P}_1} + (1 - w_i) x_i^{\mathcal{P}_2} + r_{\mathcal{P}_1} g_i^{\mathcal{P}_1} + r_{\mathcal{P}_2} g_i^{\mathcal{P}_2} \qquad \mathcal{R}_r = 1.0 \tag{4.5}$$

This recombination operator is very similar to the uniform but additionally allows to perform a simultaneous guided exploitation and separately introduces new genetic material into the population. In this context, a constant equal weight $w = 0.5$ would predominantly cause the genetic pool to rapidly converge to almost identical solutions which is likely to get stuck in suboptimal extrema and especially in multi-objective optimization. In contrast, randomized weights obtain many different subsolutions while continuously settling in potentially good regions.

4.3.5 Mutation

Mutation aims to steadily apply small changes to the genotype for which the corresponding mutation rate is typically chosen reasonably small. However, since the number of dimensions can highly vary when applied to arbitrary kinematic models, a constant value might result in either too frequent or infrequent changes. Accordingly, the mutation rate $\mathcal{M}_r$ for the mutation operator $\mathcal{M}$ denoted by (4.6) is primarily chosen inverse to the number of genes $\frac{1}{n}$. This can be interpreted as normalization of probability and is further controlled by a weight $\mathcal{M}_\alpha$. The mutation of a gene is then simply performed by adding a small value $\mathcal{M}_\beta$ that is weighted by a uniformly distributed weight w_i within $[-1.0, 1.0]$. Note that the genes are not yet clipped to their specific domain limits since further modifications are applied during the adoption phase in section 4.3.6. The parameters $\mathcal{M}_\alpha$ and $\mathcal{M}_\beta$ are both designed to ensure an adaptive variation in mutating individuals where the former represents an increase for the mutation rate and the latter a mutation strength.

$$\mathcal{M} : x_i \leftarrow x_i + w_i \mathcal{M}_\beta \qquad \mathcal{M}_r = \mathcal{M}_\alpha \frac{1}{n} \tag{4.6}$$

In order to compute $\mathcal{M}_\alpha$ and $\mathcal{M}_\beta$, an additional extinction factor ξ is calculated for each individual Ψ_i at the end of an evolution cycle according to (4.7). In this, the individuals again are assumed to be ordered by their fitness values where π_{min} relates to the best and π_{max} to the worst individual. Hence, the extinction factor is a normalized value between $[0.0, 1.0]$ where the best individual is assigned $\xi^* = 0$ and the worst $\xi^\dagger = 1$ and implicitly measures how well it has performed with respect to all other individuals within the population.

$$\xi^{\Psi_i} = \frac{\pi_i + \pi_{min}(\frac{i-1}{\psi-1} - 1)}{\pi_{max}} \tag{4.7}$$

In more detail, this factor is designed in a way such that parents whose average extinction $\xi^{\mathcal{P}_\varnothing}$ (4.8) is high will cause an overall stronger mutation within their offspring by transmitting less substantial genetic material and vice versa.

$$\xi^{P_\varnothing} = \frac{\xi^{P_0} + \xi^{P_1}}{2} \tag{4.8}$$

The mutation rate factor $\mathcal{M}_\alpha$ that adaptively controls the probability of mutation for each gene is then obtained by (4.9) and lastly normalizes the mutation rate $\mathcal{M}_r$ between $[\frac{1}{n}, 1.0]$. Ultimately, this results in a dynamic exploitation and exploration and passively encourages jumping out of local extrema in highly populated regions.

$$\mathcal{M}_\alpha = \xi^{\mathcal{P}_\varnothing}(n-1) + 1 \tag{4.9}$$

The computation of the mutation strength $\mathcal{M}_\beta$ is then performed very similar as denoted by 4.10. In this, the average extinction defines the range within the dimensional domain of the i-th gene Δ_{n_i} that can be reached by mutation. Accordingly, a higher extinction value allows an exploration over the whole search space dimension while a lower value adaptively results in an exploitation.

$$\mathcal{M}_\beta = \xi^{\mathcal{P}_\varnothing}\Delta n_i \qquad \Delta n_i = n_{i_{max}} - n_{i_{min}} \tag{4.10}$$

Note that for both operators it is possible that a single individual is picked twice as parent. In case of the fittest whose extinction factor is $\xi^* = 0$, this would result in no mutation at all. This lastly resembles a recombination rate that is slightly lower than 1.0 as indicated within the previous section 4.3.4.

This genetic mutation operator was empirically observed to perform well by neither over- nor underfitting when applied to arbitrary kinematic models. Further, the extinction factor becomes arbitrarily small for a small group of individuals that are close to local extrema while assigning larger values to other individuals in order to maintain a mutation diversity. This greatly overcomes the problem of assuming mutation parameters which might only fit well for a specific task.

4.3.6 Adoption

The adoption phase is primarily inspired by the collective behaviour of individuals who during their lifetime adopt the characteristics of their parents and aim to imitate the properties of the most successful performing prototypes within the population. This technique is very similarly used by particle swarm optimization as discussed in section 3.6 but where instead of adopting to parents, a particle relies on its own success. However, such behaviour is not exclusively present in collective systems but also takes part in natural evolution of living organisms. In mathematical terms, the adoption for each gene x_i of an offspring can mainly be realized by computing the averaged gradient to its parent genotypes $x^{\mathcal{P}_{\{1,2\}}}$ as well as the gradient to the genotype of the fittest individual x^* within the population. Both gradients are further scaled by independently evaluated uniformly distributed random values $r_{\mathcal{P}}$ and r_* between $[0.0, 1.0]$. Ultimately, a similar randomized weight w_i as for the recombination operator (4.5) is used independently for each dimension in order to maintain a dynamic balancing within adoption. The calculation of the resulting adoption operator A is given by (4.11).

$$\mathcal{A} : x_i \leftarrow x_i + w_i r_{\mathcal{P}}(\frac{x_i^{\mathcal{P}_1} + x_i^{\mathcal{P}_2}}{2} - x_i) + (1 - w_i) r_*(x_i^* - x_i) \tag{4.11}$$

This strategy can be imagined to perform a search space exploration that dynamically moves from multiple random positions that are close to good solutions towards a local extrema. Assuming a mutation that pushed the individual far away from its parents into random directions, the adoption phase can pull it back while also incorporating information of the fittest individual. This lastly allows to realize some sort of a guided random rotation around the initial position and thus is likely to improve convergence and also robustness.

Adoption is the last phase within a whole evolution cycle by which an offspring can change its characteristics and after which it has completely evolved. In order not to violate joint constraints of the inverse kinematics, all genes x_i must be clipped to their domain limits $n_{i_{min}}$ and $n_{i_{max}}$ by a function Ξ as denoted by (4.12).

$$x_i \leftarrow \Xi(x_i) \qquad \Xi : \begin{cases} x_i & n_{i_{min}} \leqslant x_i \leqslant n_{i_{max}} \\ n_{i_{max}} & x_i > n_{i_{max}} \\ n_{i_{min}} & x_i < n_{i_{min}} \end{cases} \tag{4.12}$$

Finally, the evolutionary gradient values g_i are calculated what can semantically be described by (4.13) which implies the remaining genetic change within each clipped gene x_i that has been caused during mutation and adoption. This change will then be transmitted to the offspring of the next generation for which it gives rise to a convergent direction to local extrema as indicated in section 4.3.4.

$$g_i = \Xi AMR(x_i) - R(x_i) \tag{4.13}$$

4.3.7 Niching

The niching phase is immediately performed after a new offspring Λ_i was created by its parents $\mathcal{P}_{\{1,2\}}$. In this, a preselection strategy is applied that removes any parent from the mating pool Γ who has worse fitness than its offspring. In consequence, an individual who once had success in propagating its genes is not allowed to reproduce again what passively prevents rapid accumulation of similar individuals. The corresponding niching operator $\mathcal{N}$ can be formulated as (4.14).

$$\mathcal{N} : \Gamma \leftarrow \begin{cases} \Gamma & \pi^{\Lambda_i} \geqslant \pi^{\mathcal{P}_1} \ \wedge \pi^{\Lambda_i} \geqslant \pi^{\mathcal{P}_2} \\ \Gamma \backslash \mathcal{P}_1 & \pi^{\Lambda_i} \leqslant \pi^{\mathcal{P}_1} \ \wedge \pi^{\Lambda_i} \geqslant \pi^{\mathcal{P}_2} \\ \Gamma \backslash \mathcal{P}_2 & \pi^{\Lambda_i} \geqslant \pi^{\mathcal{P}_1} \ \wedge \pi^{\Lambda_i} \leqslant \pi^{\mathcal{P}_2} \\ \Gamma \backslash \{\mathcal{P}_1, \mathcal{P}_2\} & \pi^{\Lambda_i} \leqslant \pi^{\mathcal{P}_1} \ \wedge \pi^{\Lambda_i} \leqslant \pi^{\mathcal{P}_2} \end{cases} \tag{4.14}$$

Clearly, this strategy might cause the mating pool to die out before all offspring were produced. In this case, a new offspring is created with a genotype where each gene is initialized by a uniformly distributed random value within the domain limits of its search space dimension. The niching technique is very beneficial for the robustness and the resulting convergency of the algorithm and also profits by the adaptive mutation operator that was presented in section (4.3.5). Lastly, it offers great opportunities for the exploitation phase which allows an efficient parallel search within local extrema and will be discussed in section 4.3.9.

4.3.8 Survivor Selection

As soon as all offspring Λ have evolved, the survivor selection phase is very straightforward again by using the concepts of elitism and age-based selection that were discussed in section 2.3.1. In this context, the number of offspring λ that are produced within one evolution is chosen to be $\lambda = \psi - \kappa$ where ψ is the size of the population and κ is the number of elitist individuals. The elites are simply the first κ individuals within the population Ψ which is ordered by fitness values. Hence, the survivor selection operator $\mathcal{S}_\mathcal{S}$ (4.15) is given by the union of the sets of all offspring Λ and all elites $\mathcal{K}$ which then become the new population Ψ' where $\psi' = \lambda + \kappa$ holds.

$$\mathcal{S}_\mathcal{S} : \Psi' \leftarrow \Lambda \cup \mathcal{K} \qquad \mathcal{K} = \Psi_{1,\dots,\kappa} \tag{4.15}$$

The concept of elitism becomes particularly interesting when applied to a pose objective for which the fitness function (4.3) uses a randomized weight to control the balance of position and orientation within the evolutionary target. As discussed in section 4.3.2, this randomization forces individuals to prove their performance under simulated continuously changing environments in order to ensure a dynamic search space exploration. The same also holds for the surviving elite individuals whose fitness values are recalculated with newly evaluated weights. As a direct

consequence, these elites are also frequently replaced if they do not perform highly adaptive to these changing conditions and can not maintain good fitness values.

Since the algorithm is designed for interactive inverse kinematics applications such as real-time animation and video games, a precomputation until a suitable configuration is found is mostly not desired. Instead, the so far fittest evolved individual should be applied to the kinematic model at the end of a frame. Accordingly, the fittest solution that has been found within history under the current evolutionary target must be remembered. In order to do so, it is neccessary to define a balanced fitness function $\bar{\Omega}$ which behaves similar to Ω except for a pose objective where a constant equal weighting of $w = 0.5$ is used as formulated by (4.16).

$$\bar{\pi} = \bar{\Omega}(\mathcal{X}) \qquad \bar{\Omega} : \begin{cases} d_t(\mathcal{X}, \mathcal{Y}) & \textit{Position} \\ d_r(\mathcal{X}, \mathcal{Y}) & \textit{Orientation} \\ \frac{1}{2}(\hat{d}_t(\mathcal{X}, \mathcal{Y}) + d_r(\mathcal{X}, \mathcal{Y})) & \textit{Balanced Pose} \end{cases} \tag{4.16}$$

The obtained balanced fitness value $\bar{\pi}$ can then be used by (4.17) where Φ denotes the fittest evolutionary solution and x^* is the best individual of the current population. The joint variable configuration θ is then assigned by Φ at the end of each frame or if the termination condition that is described in section 4.3.11 is fulfilled.

$$\Phi \leftarrow \begin{cases} \Phi & \bar{\Omega}(f(x^*)) \geqslant \bar{\Omega}(f(\Phi)) \\ x^* & \bar{\Omega}(f(x^*)) < \bar{\Omega}(f(\Phi)) \end{cases} \tag{4.17}$$

This aims to avoid fluctuations within the end effector when approaching, following or holding its target and the evolution underneath can continue optimization if the termination condition is not yet met.

4.3.9 Exploitation

Although it is discussed after all other evolutionary phases of the algorithm, the exploitation phase is actually the first that takes place within one evolution cycle. However, it implicitly depends on the generation that was evolved during a previous iteration since it directly operates on the elite individuals. They idea is to perform a more costly local search for the best individuals of the current population with intent to significantly increase convergency and accuracy and to guide the population. The exploitation phase is completely finished before any new offspring are created.

More particularly, an exploitation is performed on the κ fittest individuals of the population – namely all elites – by iteratively updating each gene at one time. At each iteration, a fitness value π_i is initially computed for the current genotype x by using Ω (4.3). Then, two independently evaluated uniformly distributed random values r^+ and r^- between $[0, \pm\pi_i]$ are added to the i-th gene x_i of x where

$r_- \geqslant d_{i_{min}} - x_i$ and $r_+ \leqslant d_{i_{max}} - x_i$ in order not to violate joint constraints. Accordingly, two genotypes x^+ and x^- are repeatedly obtained that are slightly modified into both domain directions at exactly one gene and equivalently give rise to π^+ and π^-. In this, the use of the fitness value as the maximum offset is reasonable since it implicitly represents a heuristic information about the change that must be applied to any joint variable in order to reach the target by the end effector. Note that for all fitness value evaluations the dynamic fitness function (4.3) with randomized weights for a pose objective is used. The update of an individual is then performed by the exploitation function $\mathcal{E}$ as formulated by (4.18). Intuitively, this strategy iteratively updates the genotype and evolutionary gradient of an individual by following the direction into which a small random heuristic offset scored a better fitness value.

$$\mathcal{E} : x_i, g_i, \pi_i \leftarrow \begin{cases} x_i, g_i, \pi_i & \pi^+ \geqslant \pi_i \ \wedge \ \pi^- \geqslant \pi_i \\ x_i + r^+, g_i + r^+, \pi^+ & \pi^+ < \pi_i \ \wedge \ \pi^+ \leqslant \pi^- \\ x_i + r^-, g_i + r^-, \pi^- & \pi^- < \pi_i \ \wedge \ \pi^- \leqslant \pi^+ \end{cases} \tag{4.18}$$

Lastly, the fitness value is averaged over the best results that were obtained for each dimension as denoted by (4.19). Such strategies are commonly used in evolutionary and collective systems since it allows to assign a more reliable stabilized fitness value to an individual that was tested under various conditions.

$$\pi = \frac{\sum_{i=1}^{n} \pi_i}{n} \tag{4.19}$$

Obviously, in this context the elite individuals are assigned a more significant relevance since they do not only control the stability but also a heuristic exploitation of the best solutions. Together with the niching technique discussed in section 4.3.7, this allows to perform in a dynamic parallel search for multiple local extrema.

4.3.10 Initialization

The initialization phase greatly influences the performance of the algorithm in terms of displacement between consecutively found solutions. Mainly, the population is initialized with individuals where all genes are computed independently for each organism by uniformly distributed random values $r_1, ..., r_n$ with respect to the domain boundaries $n_{i_{min}}$ and $n_{i_{max}}$ of the corresponding search space dimensions. However, an exclusively random initialization would permanently result in completely new solutions whenever the algorithm is restarted. Such restarts can occur if solutions are only required infrequently or if the algorithm is detected to be completely stuck in dead-end paths what will be discussed in more detail in section 4.3.12. Accordingly, exactly one individual is defined by the joint variable configuration θ that is currently assigned to the kinematic model as formulated by (4.20).

$$x^{\Psi_1} = \Big(\theta_1 \mid \theta_2 \mid \theta_3 \mid ... \mid \theta_n\Big) \qquad x^{\Psi_{2,...,\psi}} = \Big(r_1 \mid r_2 \mid r_3 \mid ... \mid r_n\Big) \tag{4.20}$$

Hence, the algorithm uses a biased initialization in order to prevent high joint variable fluctuations while niching and exploitation allows to discover new and better solutions as well as to improve the current solution simultaneously.

4.3.11 Termination

The termination condition is defined separately by the desired Cartesian accuracy in position and orientation of the end effector to its target. Let $E^{\mathcal{Y}}_{\{P,O\}}$ denote the maximum translational and rotational errors for termination and $E_{\{P,O\}}$ the current errors obtained from the solution Φ to the evolutionary target $\mathcal{Y}$ using d_t (2.13) and d_r (2.14), so the termination condition can be formulated as (4.21).

$$\begin{aligned} &E_P = d_t(f(\Phi), \mathcal{Y}) \quad E_O = d_r(f(\Phi), \mathcal{Y}) \\ &E_P \leqslant E^{\mathcal{Y}}_P && \textit{Position} \\ &E_O \leqslant E^{\mathcal{Y}}_O && \textit{Orientation} \\ &E_P \leqslant E^{\mathcal{Y}}_P \ \wedge \ E_O \leqslant E^{\mathcal{Y}}_O && \textit{Pose} \end{aligned} \tag{4.21}$$

This arbitrary choice of accuracy is greatly beneficial for interactive applications in which precise solutions are less important than fast computation times. In this context, it ensures to stop the evolution from further unneccessary optimization and to continue only when required.

4.3.12 Wipe

Although the algorithmic design allows a very robust and dynamic search space exploration, situations can occur in which all elites and most other individuals have settled within numerically good suboptimal extrema. In those, it is usually neccessary to perform a large instantaneous mutation in order to discover new local extrema and to escape from the current. However, the discovered genotype would be required to immediately score a lower fitness value than any other within the currently known suboptimal regions. As a consequence, the probability to directly mutate into a better region becomes increasingly low and causes the population to be stuck in exploiting dead-end paths. In order to avoid such situations and thus to significantly increase robustness, the wipe criterion (4.22) at the end of one evolution cycle heuristically determines whether a new initialization of the whole population is required. The heuristic first checks whether the last evolved generation can score progress on the current solution Φ by comparing it to the fittest individual x^* with respect to the balanced fitness function $\bar{\Omega}$ (4.16). Second, if no exploitation (4.18) for any gene of x^* was successful, the population is wiped and reinitialized according to (4.20). The initialization phase then takes care of false positive wipes by integrating the current solution into the population.

$$\bar{\Omega}(f(\Phi)) < \bar{\Omega}(f(x^*) \ \wedge \ \nexists i : \bar{\Omega}(f(\mathcal{E}_i(x^*))) < \bar{\Omega}(f(x^*)) \tag{4.22}$$

4.4 Résumé

The algorithmic approach was developed to be applicable to arbitrary kinematic models where the evolution adaptively reacts to the current optimization progress and internal state of the population. More particularly, the intent in the designed operators, modifications and extensions is to prevent overfitting and to remain efficient with respect to the performance measures that were defined in section 4.1.

While related approaches usually require many parameters, only the size of the population ψ and the number of elites κ need to be specified. However, these are quite straightforward since there should always hold the relation $\kappa \ll \psi$. The challenge remains in finding a good balance in the synergy of the raw evolutionary optimization and the comparatively costly exploitation.

Further, both the rate and strength parameters of the mutation operator are not controlled by any predefined values but instead are determined adaptively. In this, the introduced extinction factor allows to remain sensitive to local extrema since it can grow arbitrarily small. Simultaneously, it encourages genetic diversity by performing a stronger and more frequent mutation in offspring of less successful parents and thus prevents the emergence of highly overpopulated regions.

The objective function uses biologically plausible random weights and hence avoids to be tuned individually for various kinematic models. In addition, it ensures a dynamic search space exploration that improves convergency and also allows to escape from suboptimal extrema since fitter individuals must repeatedly prove successful under various conditions. Also, it remains scale-independent in case of a pose objective due to a rebalanced translational distance function.

The dynamic exploitation of the fittest individuals further allows an efficient refinement of potentially good solutions. This technique also profits from the integrated niching strategy and positively affects the whole evolutionary optimization. In more detail, the exploitation can search within multiple local extrema simultaneously and thus improves the convergency, robustness and flexibility in finding inverse kinematics solutions.

The computation of evolutionary gradients heuristically aims to transmit genetic improvements within parents to their offspring during recombination. The further biologically plausible adoption phase then results in a continuous omnidirectional exploitation around local extrema.

A great advantage in contrast to pure gradient search strategies lastly comes with the wipe heuristic. While those in case of random restarts typically require to start from scratch and result in completely new solutions, the presented algorithm can keep track of the current and simultaneously discover new and better solutions.

All together, the pseudocode of the complete algorithm is depicted in Fig. 4.2 and gives a detailed description for implementation. In addition, a maximum amount of time is specified that implicitly controls the number of generations that can be performed within one frame. This is important in order to meet the requirements for interactive applications where constant high frame rates must be maintained.

```
1  Parameters: ψ, κ;
2  Initialize population;
3  ⟩⟩ Evaluate fitness of all individuals;
4  ⟩⟩ Sort population by fitness;
5  ⟩⟩ Calculate extinction factors;
6  while termination condition not satisfied do
7      Exploit κ fittest individuals;
8      Assign whole population to the mating pool;
9      for ψ − κ do
10         if mating pool is not empty then
11             Select two individuals from mating pool as parents;
12             Create individual by Recombination, Mutation and
                Adoption and perform Clipping;
13             Calculate evolutionary gradient;
14             Evaluate fitness;
15             Remove worse parents from mating pool;
16         else
17             Create random individual;
18             Evaluate fitness;
19         Add individual to offspring;
20     Select κ elites and all offspring to constitute the new population;
21     Sort population by fitness;
22     Calculate extinction factors;
23     Update evolutionary solution;
24     if wipe criterion fulfilled then
25         Reinitialize population;
26     if frame time limit exceeded then
27         Assign joint variable configuration;
28         Wait for next frame;
29 Assign joint variable configuration;
```

Figure 4.2: Pseudocode of the Hybrid Genetic Swarm Algorithm

Chapter 5

Experimental Analysis

"No amount of experimentation can ever prove me right; a single experiment can prove me wrong." – Albert Einstein

This chapter examines the capabilities of the Hybrid Genetic Swarm Algorithm that was presented in detail in section 3.

Initially, section 5.1 describes the environmental setup for the experiments and section 5.2 introduces the primarily used kinematic models.

The experimental analysis is then mainly subdivided into four parts where each evaluates different aspects of the algorithmic solution.

First, section 5.3 performs a parameter study in order to demonstrate the effects of the individual parameters as well as to determine a reasonable parameter choice which obtains a good algorithm efficiency.

Second, a selective study with respect to the introduced algorithmic improvements and modifications is conducted within section 5.4 and especially demonstrates the improvement ratio over simple genetic algorithms.

Section 5.5 follows with a performance study to show the efficiency of the complete algorithmic solution for various kinematic models regarding the introduced performance criteria that involve success, accuracy, time, displacement and flexibility.

Lastly, section 5.6 performs a comparative study to other related methods that were published very recently and aims to highlight resulting opportunities and limitations of the algorithm.

5.1 Environmental Setup

The evaluation setup (Fig. 5.1) of the presented algorithm was implemented in C# and Unity (Version 5.3.1f1 Personal) [81] using an ASUS G751JY-T7159H notebook. The implementation runs on a single processor core with up to 3.4GHz. A URDF (Unified Robot Description Format) parser was implemented in order to import common robot models and to load their specific kinematic geometry. Following the modular software design architecture of Unity, two script components were implemented. The first models a kinematic joint and the second solves the inverse kinematics problem on an arbitrary hierarchical kinematic chain.

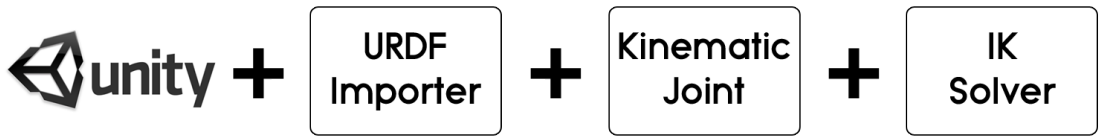


Figure 5.1: Implementation Setup in Unity

The kinematic joint editor component is illustrated in Fig. 5.2 and allows to specifiy the either translational or rotational movement where each motion axis can independently be adjusted by lower and upper limits as well as a maximum velocity and acceleration in order to achieve realistic motions. Note that zero values for velocity and acceleration are interpreted to immediately teleport the connected links at the desired joint values.

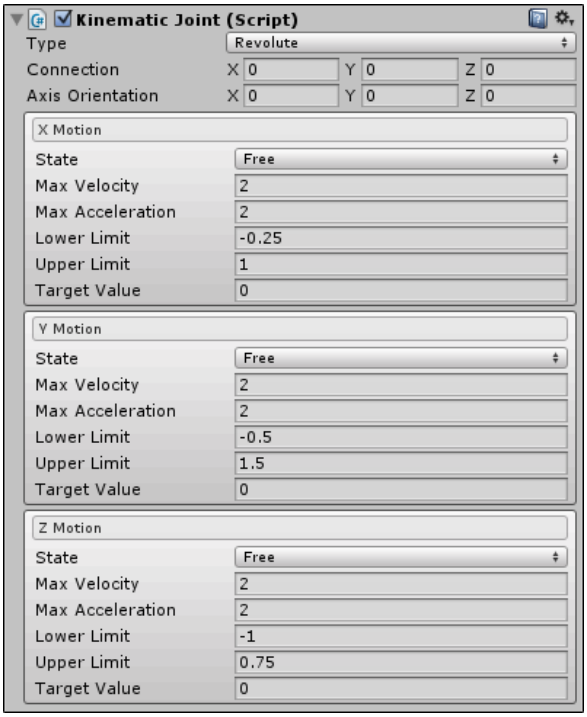


Figure 5.2: Kinematic Joint Editor Component

The inverse kinematic solver editor component shown in Fig. 5.3 is then added to the end effector and automatically determines the hierarchical composition of kinematic joints and builds a kinematic model to obtain the forward kinematics. The target is given by a Transform of an arbitrary GameObject and is tracked by the inverse kinematics algorithm. In order to ensure interactive frame rates, a maximum frame time is specified to implicitly constrain the maximum number of generations between two consecutive frames. Further, the Cartesian objective, the maximum allowed errors for termination and also the algorithm parameters can be set. Lastly, the implementation allows to ignore specific joints if some movements shall be explicitely restricted or in case of multple end effectors where identical joints are contained in several kinematic chains but shall only be used by a certain end effector.

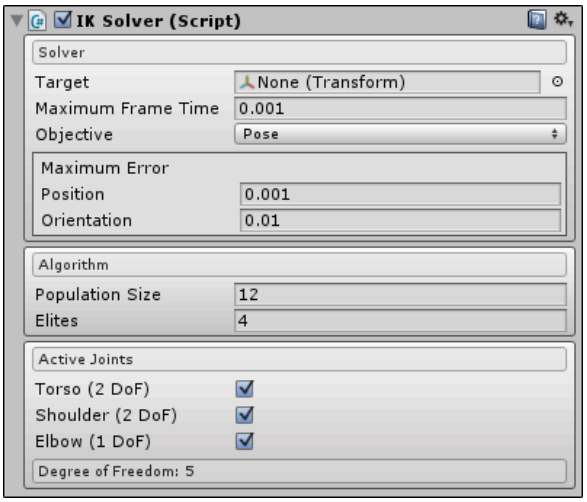


Figure 5.3: IK Solver Editor Component

5.2 Kinematic Models

For the experiments, several kinematic models with different geometry and with varying degree of freedom from 6 up to 25 were used. Details are shown in Fig. 5.4 and Tbl. 5.1. The models were either loaded using the implemented URDF importer for which the kinematic specifications were directly applied or were imported and specified manually (Kyle & Dragon). The chosen models originate from different areas of application such as industrial manufacturing as well as humanoid and service robots and also video game characters. The kinematic chain for evaluation is illustrated by the consecutive set of joint axes where each independently defines one axis of motion.

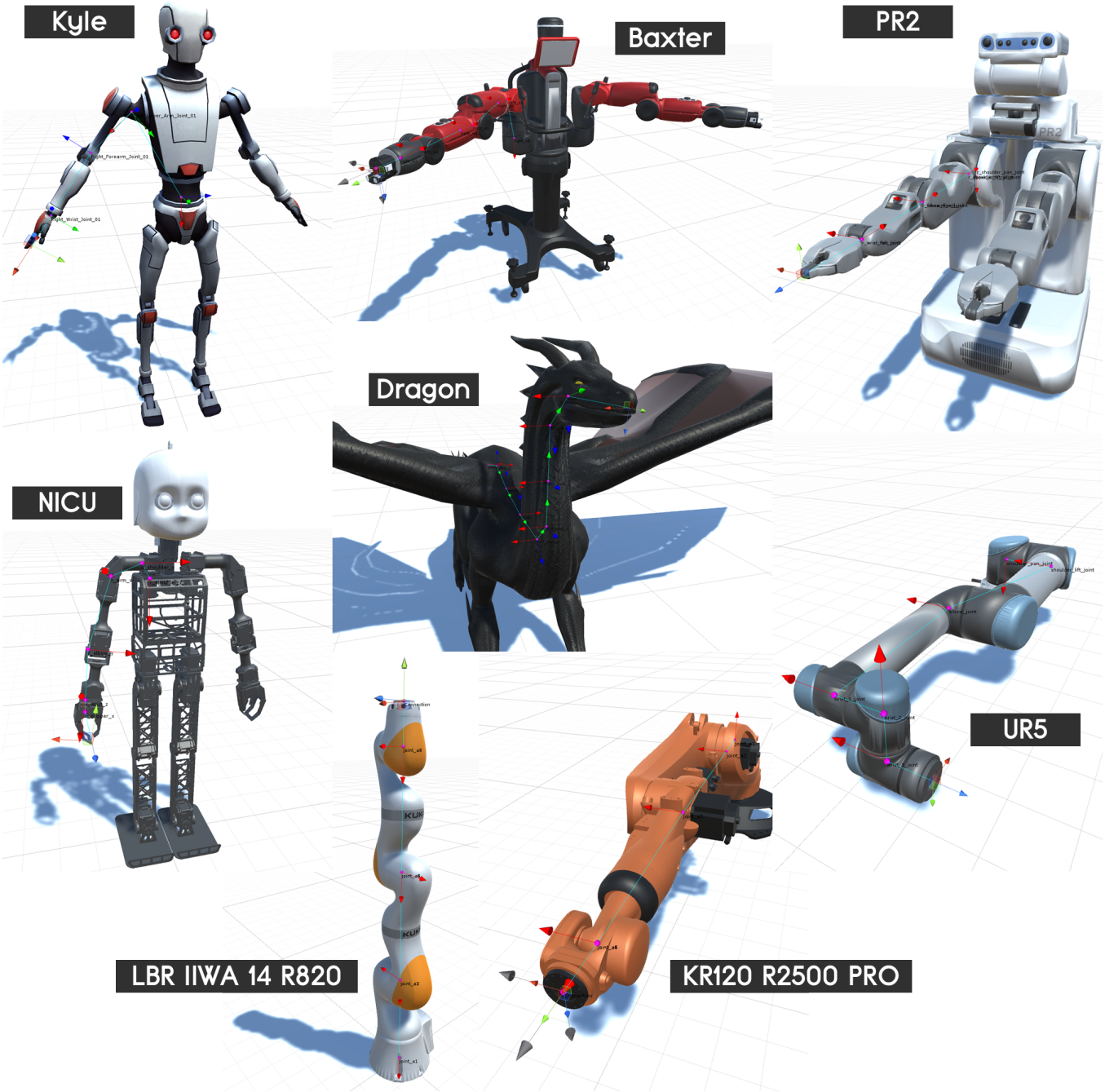


Figure 5.4: Kinematic Models

Model	Kyle (Pelvis+Arm) [47]	Dragon (Pelvis+Neck) [22]	PR2 (Arm) [66]	UR5 [82]	NICU (Arm) [83]	LBR IIWA 14 R820 [68]	KR120 R2500 PRO [67]	Baxter (Arm) [6]
DoF	10	25	7	6	6	7	6	7

Table 5.1: Degree of Freedom of Kinematic Models

5.3 Parameter Study

As mentioned in chapter 4, the only parameters of the algorithm that are required to control the evolutionary optimization are given by the population size and the number of elite individuals. Usually, the population size in genetic algorithms is chosen relatively high in contrast to the amount of elites within the population. Recalling that the algorithm shall be able to run at high frame rates where each evolutionary cycle requires comparatively few computation time, a small population size is preferred. The challenge then becomes to achieve a fast and robust convergence within a short time window.

In this context, a parameter study was conducted on the UR5 manipulator where Fig. 5.5 and Fig. 5.6 demonstrate the influence of both parameters with respect to the averaged gain in fitness improvement and computation time increase over a fixed amount of 100 generations. Obviously, treating all individuals as elites – and thus performing local searches on the whole population as described in section 4.3.9 – will yield fitter individuals within same generations. However, using a few more elites also causes a much higher gain in computation time than for increasing the population size. This is due to the exploitation requiring many more fitness evaluations than a newly created offspring where only one calculation is needed. In

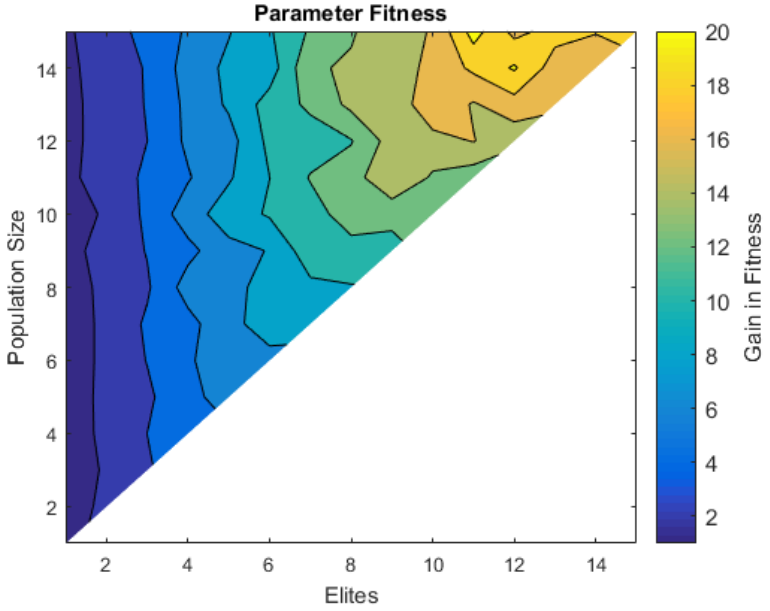


Figure 5.5: Fitness Gain of Parameters

order to achieve an efficient optimization, a reasonable balancing of the population size and the number of elites is required. Accordingly, Fig. 5.7 demonstrates the normalized parameter efficiency that is obatined by incorporating information of the combined optimum of both derivatives in fitness and computation time gain. The analysis suggests to choose the population size two to three times as high as the

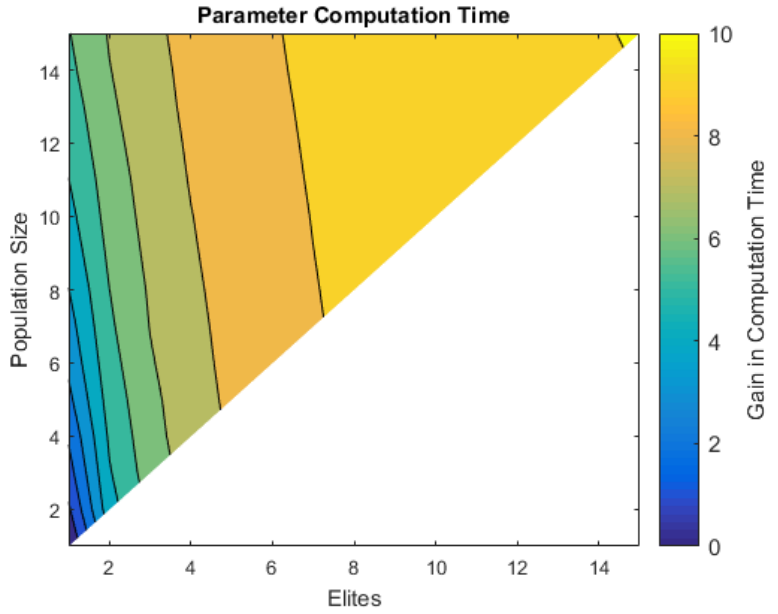


Figure 5.6: Computation Time Gain of Parameters

number of elites. This could be confirmed by empirical results which further also showed similar performance for same parameter selections under various kinematic models. Hence, the parameter selection that is used during the next sections is chosen to be $\psi = 12$ and $\kappa = 4$. Although the population size could also have been chosen appropriately smaller, a larger amount of individuals has shown to provide a higher robustness as well as smaller standard deviation in convergency.

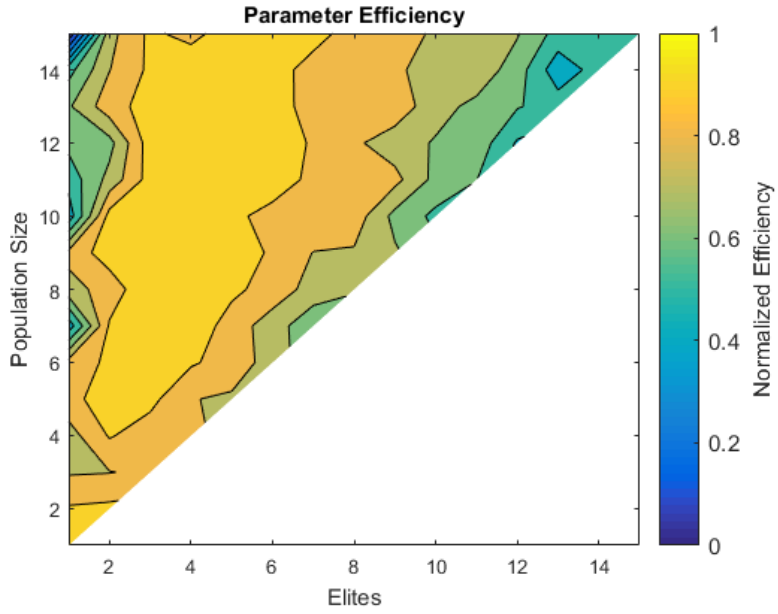


Figure 5.7: Parameter Efficiency

5.4 Selective Study

This section aims to specifically demonstrate the improvements that could be obtained by the introduced algorithmic extensions and modifications. The experiments are conducted using the kinematic model of the UR5 robot arm. First, section 5.4.1 shows the overall improvement in convergency of the complete Hybrid Genetic Swarm Algorithm compared to simple genetic algorithms. Section 5.4.2 then confirms the intended effects of the designed extinction factor in maintaining genetic diversity as well as sensitivity to local extrema. The following sections 5.4.3 - 5.4.6 then individually show the loss in efficiency that would occur if either multi-objective weight randomization, adoption, exploitation or the wipe criterion were removed from the complete algorithm. All experiments except those in section 5.4.2 are conducted within a limited time frame of $\frac{1}{30}$s over 10.000 randomly generated reachable poses derived from valid joint variable configurations. In those, the obtained curves show the mean as well as standard deviation values of the currently evolved solution over all samples at each discrete time step. The optimization is always started from the default posture of the kinematic model.

5.4.1 HGSA versus SGA

The most representative criteria to measure the efficiency of the algorithm can be defined by a continual convergency to an optimal solution. Accordingly, Fig. 5.8 shows the fitness convergency of HGSA over SGA where the results indicate that HGSA is able to significantly outperform SGA both in terms of mean and standard deviation measurements. More particularly, SGA struggles in scoring

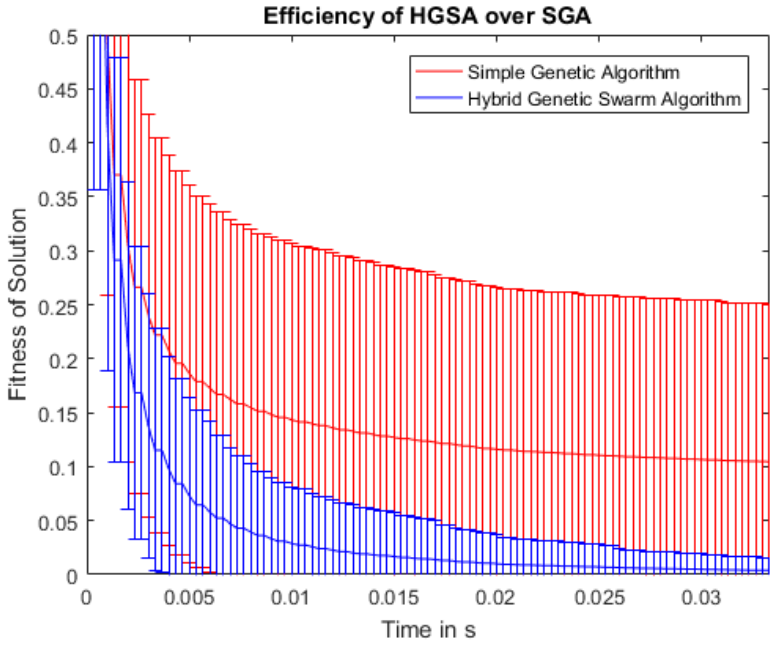


Figure 5.8: Fitness Convergency of HGSA over SGA

further progress by reaching the specified time limit although a rapid initial improvement of the solution as for HGSA could be achieved. The fitness convergency is explicitely measured relative to the elapsed time than to the number of evolved generations since the computation time for one evolutionary cycle obviously depends on the algorithmic design. In more detail, one generation for SGA is much cheaper than for HGSA since fewer fitness evaluations and heuristic calculations are required where relying on the amount of generations would not give qualitative results for measuring the algorithmic efficiency. Further, Fig. 5.9 illustrates the improvement ratio of HGSA over SGA that can be derived from Fig. 5.8 and which can be observed to maintain a steadily growing increase in efficiency.

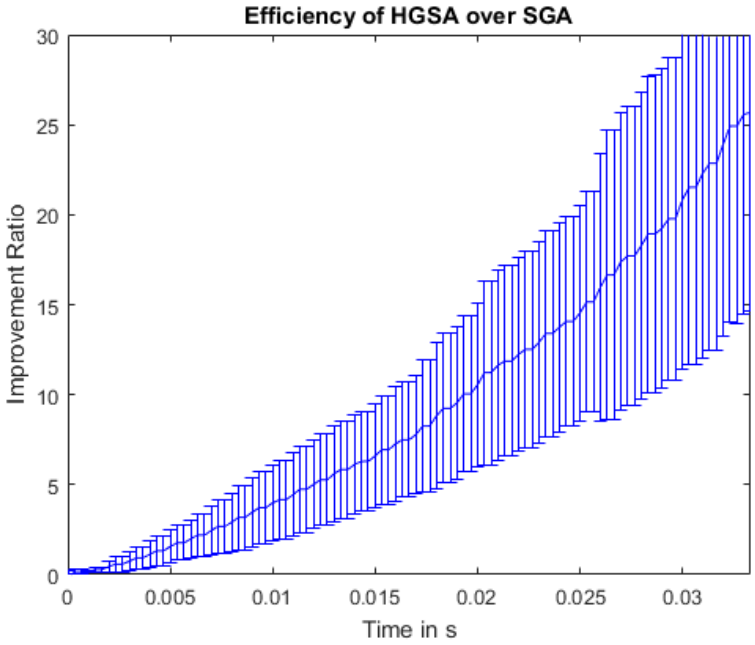


Figure 5.9: Improvement Ratio of HGSA over SGA

5.4.2 Extinction Factor

As described in section 4.3.5, the extinction factor was designed to let the population itself determine the required amount of exploitation and exploration in mutation since fixed values for mutation rate and strength would not be suitable for arbitrary kinematic models of varying geometric complexity. The evolution of the extinction factors over 100 generations within a population of 12 individuals is visualized by Fig. 5.10. Considering the fitness convergency over generations, it can be observed that the individual extinction factors dynamically adapt to the current evolutionary progress. More particularly, the extinction factor of the best individual within the population constantly remains zero and – if chosen as parent – will try to maintain the characteristics within its offspring. In contrast to this, the worst individual will tend towards a complete mutation of the whole genotype

within its offspring. All individuals in between adaptively react by maintaining explorative diversity and ensuring sensitivity to local extrema. With effect of the parent selection operator as described in section 4.3.3, the outcome is a reasonable and dynamic mutation rate with respect to the degree of freedom as well as a suitable mutation strength that can grow arbitrarily small in order to scale with the desired solution accuracy.

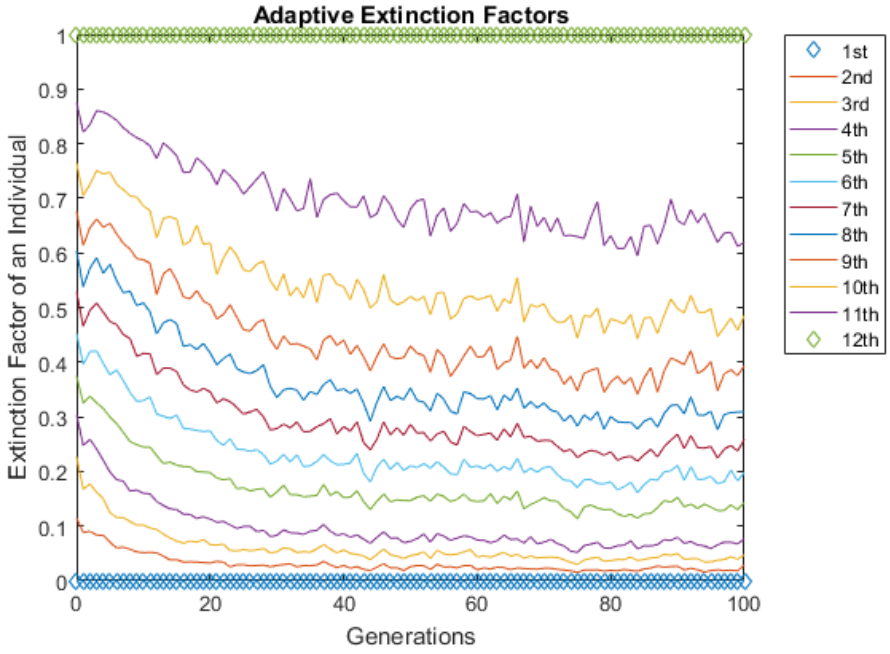


Figure 5.10: Evolution of Extinction Factors over 100 Generations

5.4.3 Multi-Objective Weight Randomization

The idea of using randomized weights for the multi-objective optimization problem to satisfy a pose objective of individually weighted position and orientation distances was introduced in section 4.3.2 and was inspired by Darwin's statement on the fittest individuals which are "most responsive to change". Obviously, using randomized instead of specifically tuned weights is from an algorithmic perspective much simpler and at the same time induces a significant increase in convergency and especially robustness as depicted in Fig. 5.11. While equal weights seem to get stuck in suboptimal extrema by suddenly slowing down convergency and stop scoring any further progress, randomized weights show a slightly less greedy behaviour by initial improvements but are able to maintain a continual fitness convergency after very few milliseconds. From a computational perspective, this is a very strong result since randomized weights require no considerable additional cost but create a highly dynamic and beneficial search space exploration for a mathematically complex problem. Fig 5.12 shows the related improvement ratio that can be obtained by using multi-objective weight randomization.

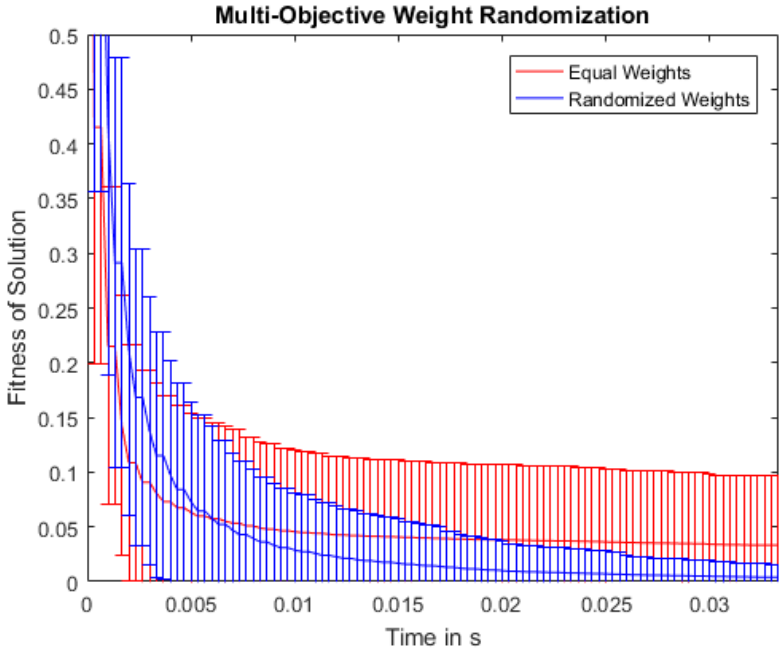


Figure 5.11: Effect of Multi-Objective Weight Randomization in Fitness Convergency

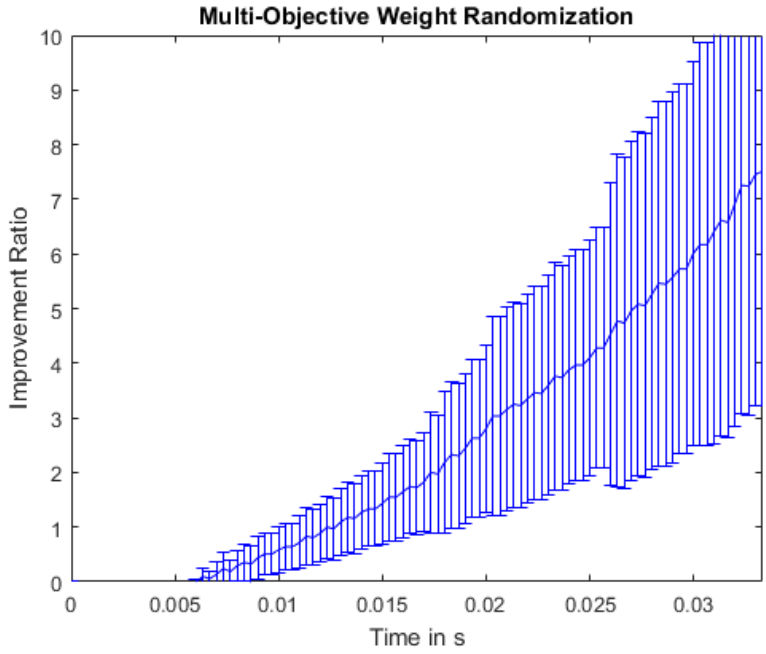


Figure 5.12: Improvement Ratio by Multi-Objective Weight Randomization

5.4.4 Adoption

The adoption phase was described in section 4.3.6 and imitates the collective behaviour of organisms as observed in natural phenomena and aims to let offspring adopt the characteristics of their parents and the most successfully performing individual among the current population. More particularly, the motivation was to create constant system dynamics similarly to swarm intelligence and to increase convergency as well as to encourage exploitation around local extrema. In this sense, Fig. 5.13 confirms the intended effects in efficiency where adoption accomplishes a continuously faster fitness convergency within same time frames as if no adoption was applied. The resulting improvement ratio for the algorithmic ap-

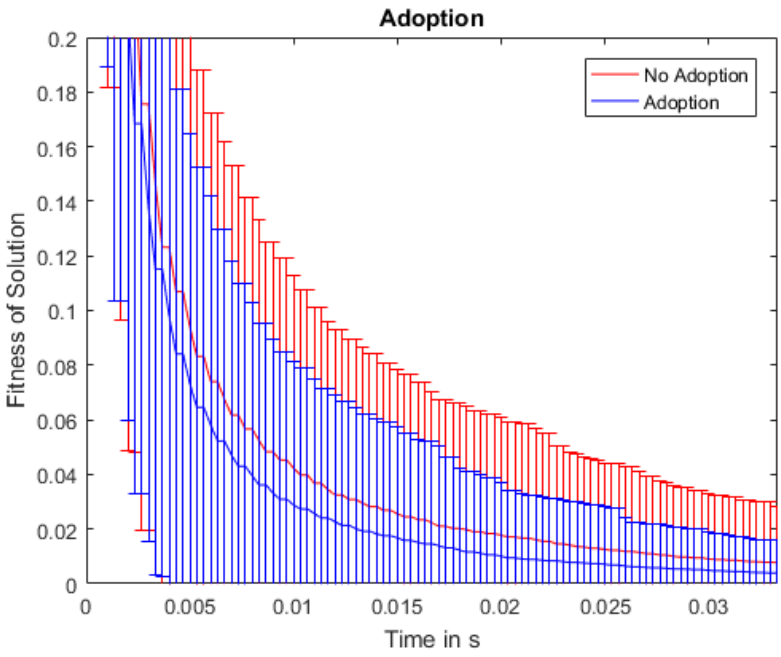


Figure 5.13: Effect of Adoption in Fitness Convergency

proach is depicted in Fig. 5.14 and shows a slightly sublinear increase with respect to the elapsed time. Considering the factorial efficiency gain of this improvement under the presence of all other introduced and involved algorithmic extensions, the adoption phase can contribute to a highly beneficial speedup during the whole evolutionary optimization. This is particularly interesting from an algorithmic perspective in terms of biologically-inspired optimization strategies where the results suggest that a combination of genetic algorithms and particle swarm optimization can robustly achieve an increase in computational efficiency by a set of simple heuristics. The combination of these two optimization strategies is a very modular and generic hybridization which reinforces the hypothesis that biologically plausible concepts are able to realize more efficient and sophisticated computational methods.

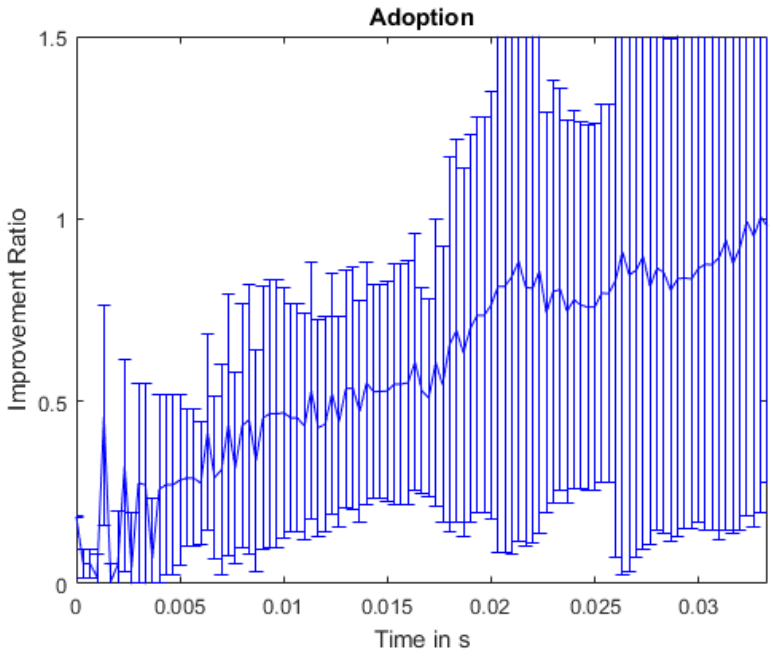


Figure 5.14: Improvement Ratio by Adoption

5.4.5 Exploitation

The exploitation technique was introduced in section 4.3.9 and aims to explicitely search for improvements within the genetic material of elite individuals. The intent of this extension was to achieve a significant increase in convergency especially for already precise solutions and to implicitly guide the search space exploration of the whole population. Accordingly, Fig. 5.15 shows the noteable improvement in fitness convergency when performing an exploitation on the fittest individuals among the population. The observable progress by the mean and standard deviation values behaves very similar as for the effects of adoption and contributes to a continual improvement during the whole evolutionary optimization. However, the gain of the algorithmic improvement ratio rises much faster and shows an approximately linear increase with respect to the elapsed time in optimization as depicted in Fig. 5.16. Obviously, the exploitation phase together with the multi-objective weight randomization contribute the largest impact in fitness convergency within the specified time frame, but jointly accomplish an even more significant and steady speedup over simple genetic algorithms in correlation with the adoption phase. In more detail, the exploitation technique and adoption phase are especially relevant for a rapid initial improvement of the solution what can be observed by examining the effects within the first five milliseconds in Fig. 5.12, Fig. 5.14 and Fig. 5.16. Equivalently with respect to very precise solutions at further runtimes, a huge improvement is scored by the interaction of multi-objective weight randomization and the exploitation of elites.

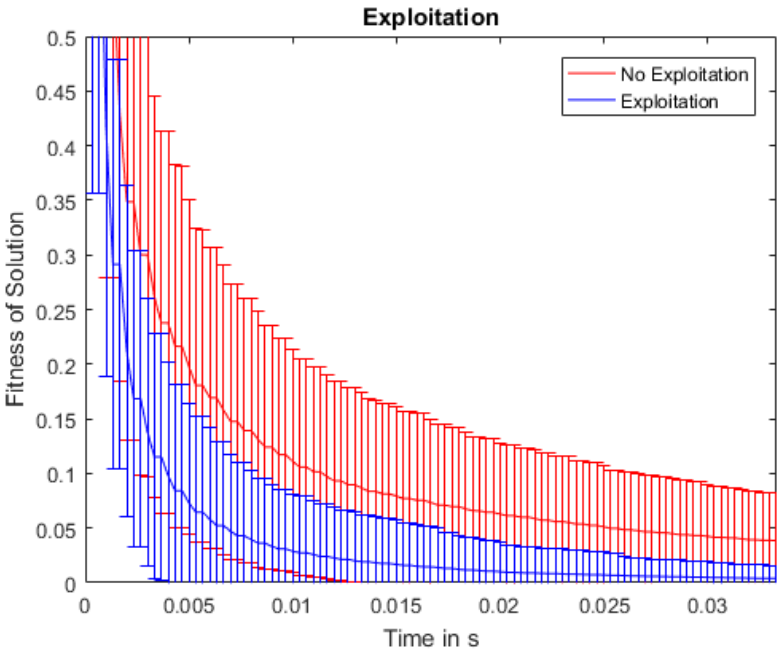


Figure 5.15: Effect of Exploitation in Fitness Convergency

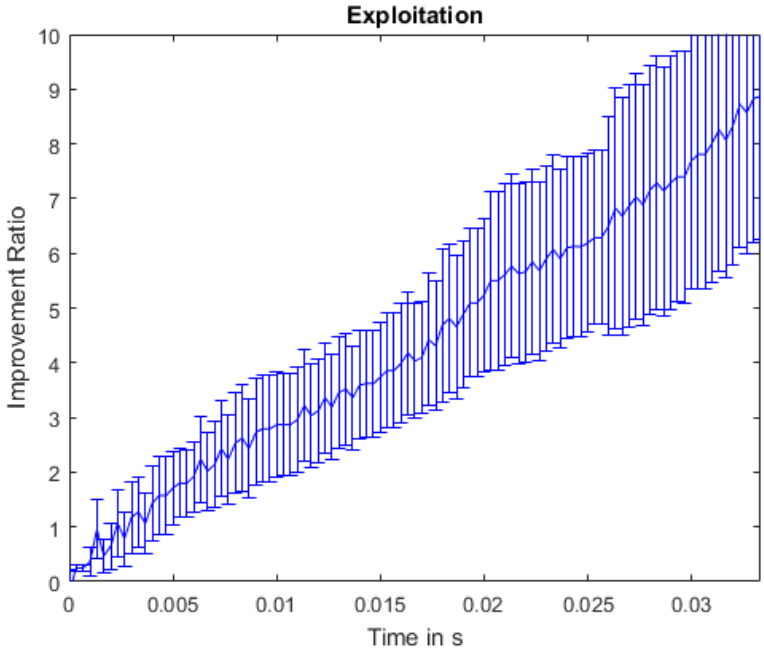


Figure 5.16: Improvement Ratio by Exploitation

5.4.6 Wipe Criterion

The wipe criterion in section 4.3.12 was especially designed for kinematic models with a high amount of local extrema within search space and where situations might occur in which the whole population has entirely settled within suboptimal extrema configurations. In those, a simple restart of the algorithm might be more efficient than to continue evolution until the population can successfully escape from dead-end paths. However, the challenge remains in a reliable detection of such situations and thus not to unneccesarily slow down the whole optimization progress. As depicted in Fig. 5.17, the algorithm can successfully avoid getting stuck in suboptimal extrema what can be observed if no wipe of the population is allowed. At the same time, the fitness convergency does not seem to be slowed down during the initial improvement of the solution within the first few milliseconds. This clearly indicates the reliable detection of an exclusive suboptimal extrema exploitation and what can be confirmed by Fig. 5.18 which visualizes the related improvement ratio that is achieved by the wipe criterion. In this, the wipe criterion

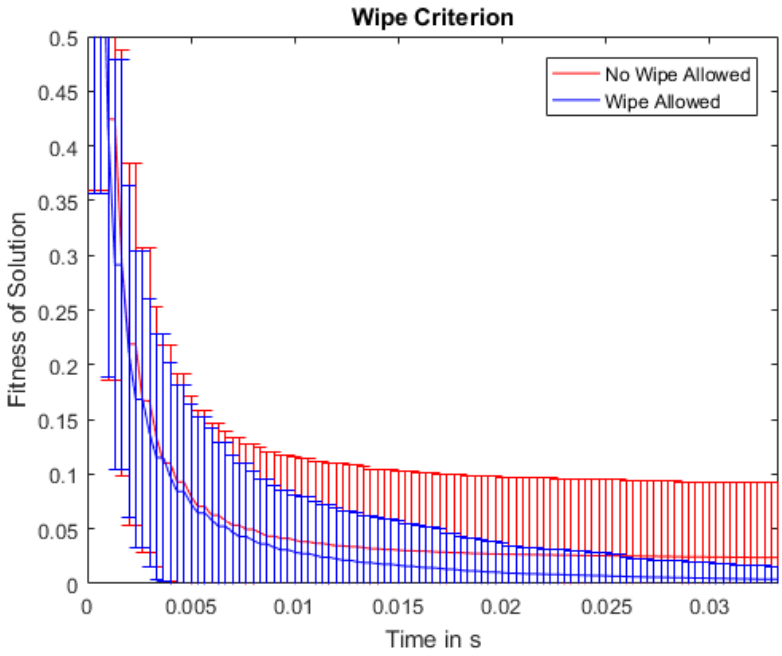


Figure 5.17: Effect of the Wipe Criterion in Fitness Convergency

immediately scores significant improvement after five milliseconds which coincides with the timestamp where the algorithm starts slowing down in fitness convergency if no wipe is allowed. Overall, a huge increase within the improvement ratio can be observed what is particularly caused by a higher probability of falling into the global search space optimum by partially reinitializing the optimization where further progress in accuracy can be scored.

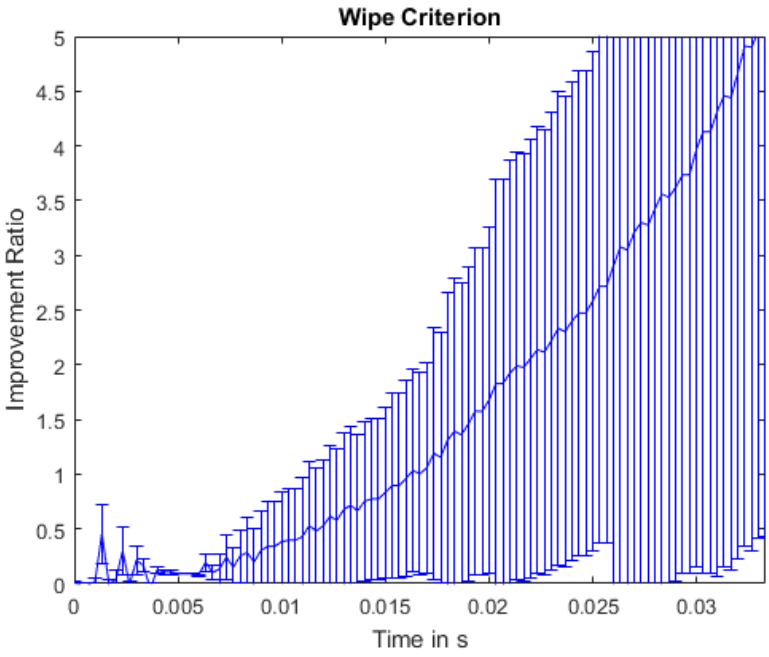


Figure 5.18: Improvement Ratio by the Wipe Criterion

5.5 Performance Study

This section examines the overall efficiency that can be achieved by the Hybrid Genetic Swarm Algorithm regarding the previously introduced performance criteria in section 4.1. First, section 5.5.1 demonstrates the success rate in finding suitable inverse kinematics solutions. Sections 5.5.2 and 5.5.3 then study the related tradeoff in the desired accuracy of position and orientation with respect to the required computation time. The attainable displacement between consecutive solutions is then examined in section 5.5.4. Lastly, section 5.5.5 shows the algorithmic performance in flexibility regarding a greatly increasing degree of freedom as well as varying joint types. All experiments in this section are conducted on reachable target configurations of either 10.000 randomly generated samples (if no further specified) or continuous trajectories. The search queries on independent targets are started from the default posture of the kinematic models while tracking experiments continued approximation from the previously evolved solution.

5.5.1 Success

The success rate in finding a theoretically existent solution is one of the most important properties in optimization algorithms. Obviously, the measured success implicitly depends on the desired solution accuracy for which the maximum errors in position and orientation were chosen reasonably small with 0.001m and 0.01rad. More particularly, a reasonably high accuracy typically indicates to have found the global optimum where further progress only depends on the available computation

time. Fig. 5.19 visualizes the satisfied target poses over 1000 random samples for three different kinematic models of highly varying kinematic geometry and complexity. In this, all solutions could successfully be evolved. Equivalent results can be observed by Fig. 5.20 for 8000 uniformly distributed samples under a position objective.

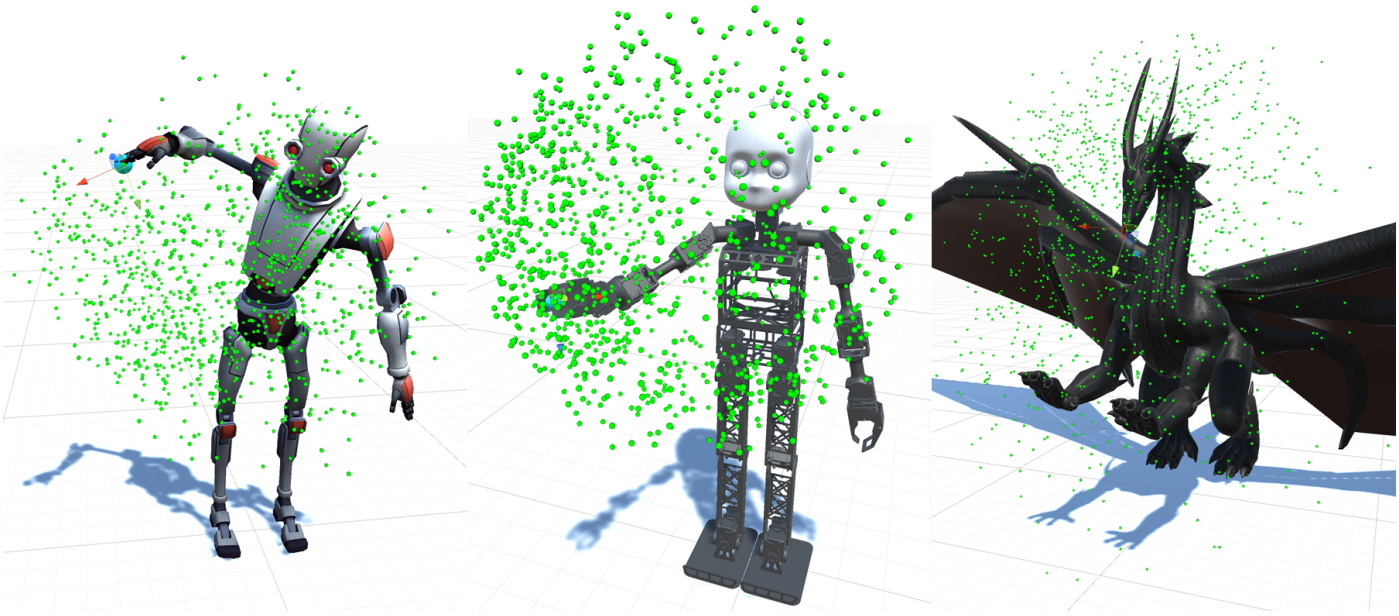


Figure 5.19: Successfully satisfied Target Poses over 1000 Random Samples

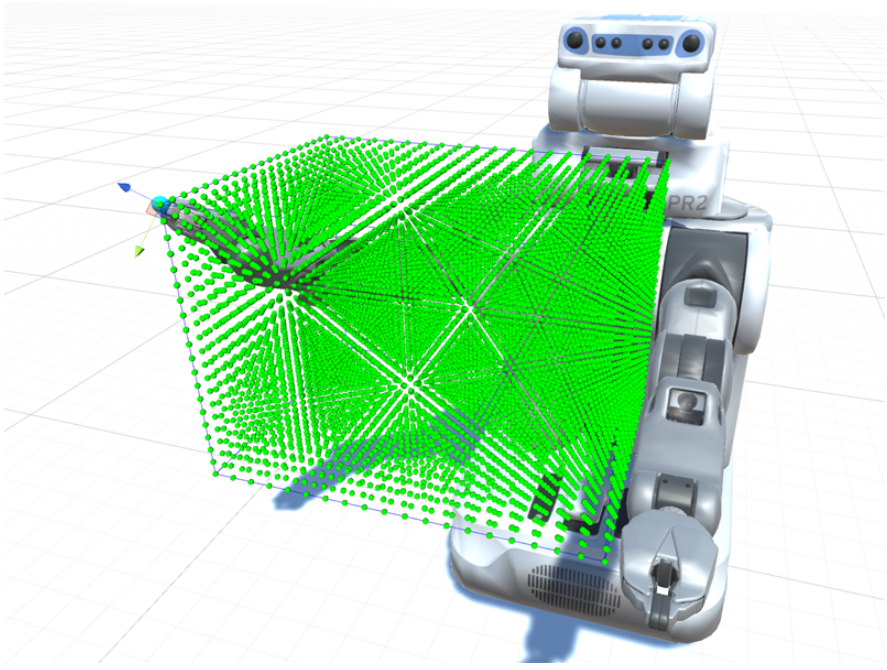


Figure 5.20: Successfully satisfied Target Positions over 8000 Uniform Samples

In order to demonstrate the success rate of the proposed algorithm, similar experiments on 10.000 randomly generated target poses were conducted for various kinematic models. The results are depicted in Fig. 5.21 and show that a success rate of 100% could be obtained for each kinematic model by a certain number of generations. From an algorithmic perspective, this experimental result particularly shows the performance capabilities in search space exploration and convergency regardless of the underlying computational power. It can be observed that the average number of generations until any solution is found is typically located somewhere around 10^2 depending on the complexity of the kinematic model.

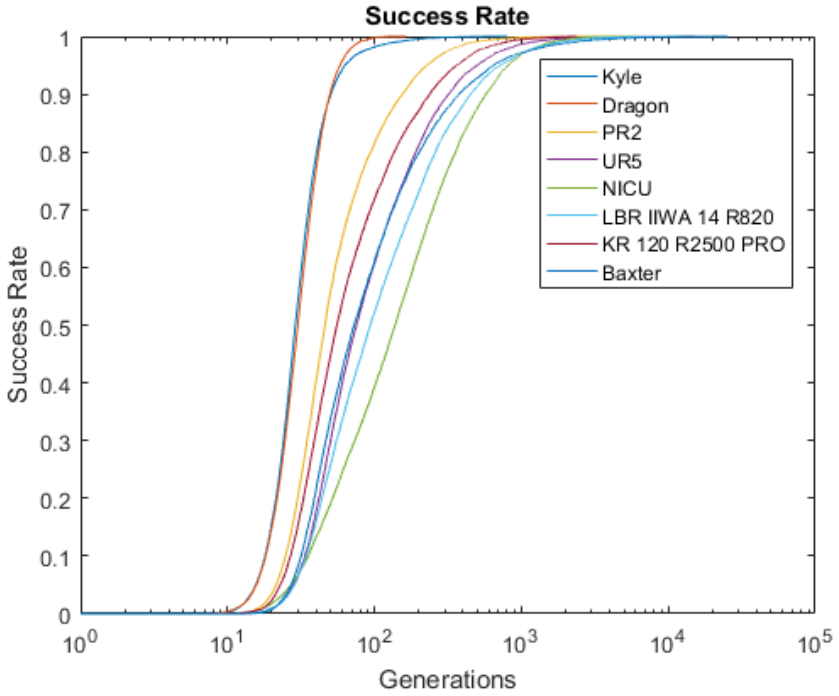


Figure 5.21: Success Rate under a Pose Objective over 10.000 Random Samples

In this context, further experiments regarding the related mean and standard deviation values in the expected number of generations to evolve a solution under a pose objective were performed. The results for the specific kinematic models are shown in Fig. 5.22. Interestingly, although the models Kyle and Dragon have a comparatively high degree of freedom, the results in the required amount of generations both in mean and standard deviation are surprisingly low. In further observations, although the degree of freedom among the remaining models is almost identical, the variation in generations shows significant differences. Implicitly, this allows to make assumptions about the complexity of the kinematic model and its underlying search space and thus the amount of suboptimal extrema configurations and singularities that must be surmounted by the evolutionary optimization.

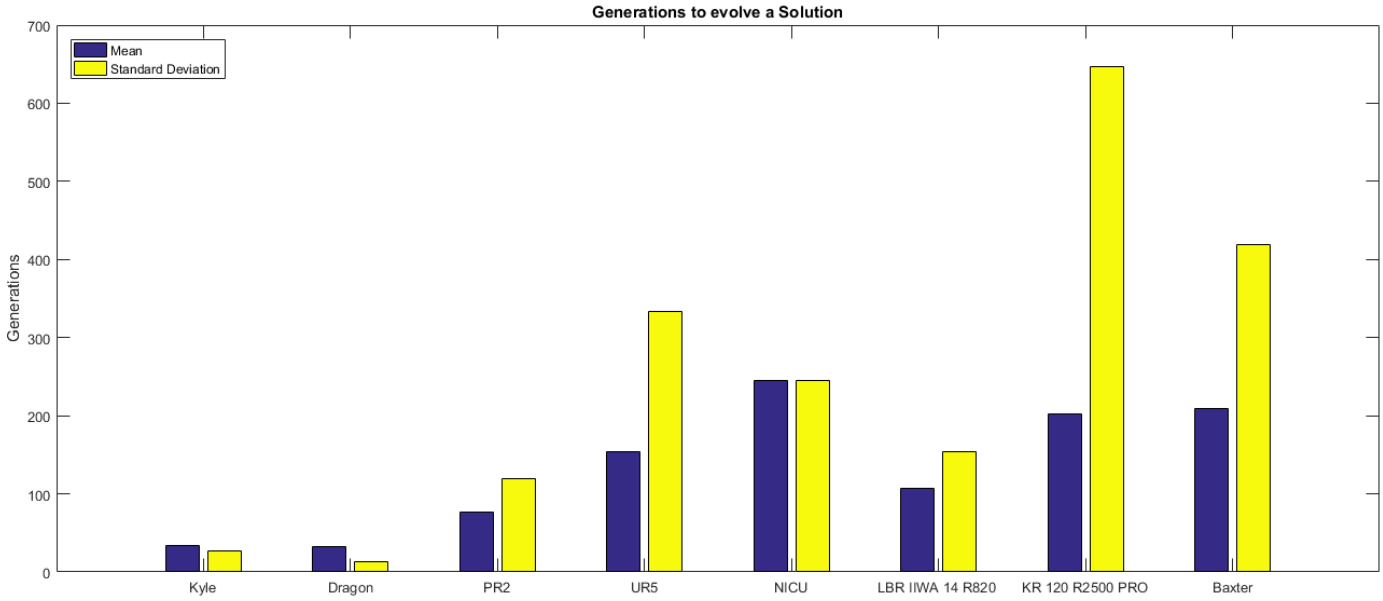


Figure 5.22: Number of Generations to evolve a Solution under a Pose Objective

5.5.2 Accuracy

The attainable accuracy of the algorithm was evaluated both for random target poses as well as by following a continual trajectory of consecutive and directly neighbouring Cartesian configurations. The former experiment was conducted on multiple kinematic models for which the results are shown in Fig. 5.23 where a maximum time frame of $\frac{1}{30}$s was specified. It can be observed that the mean

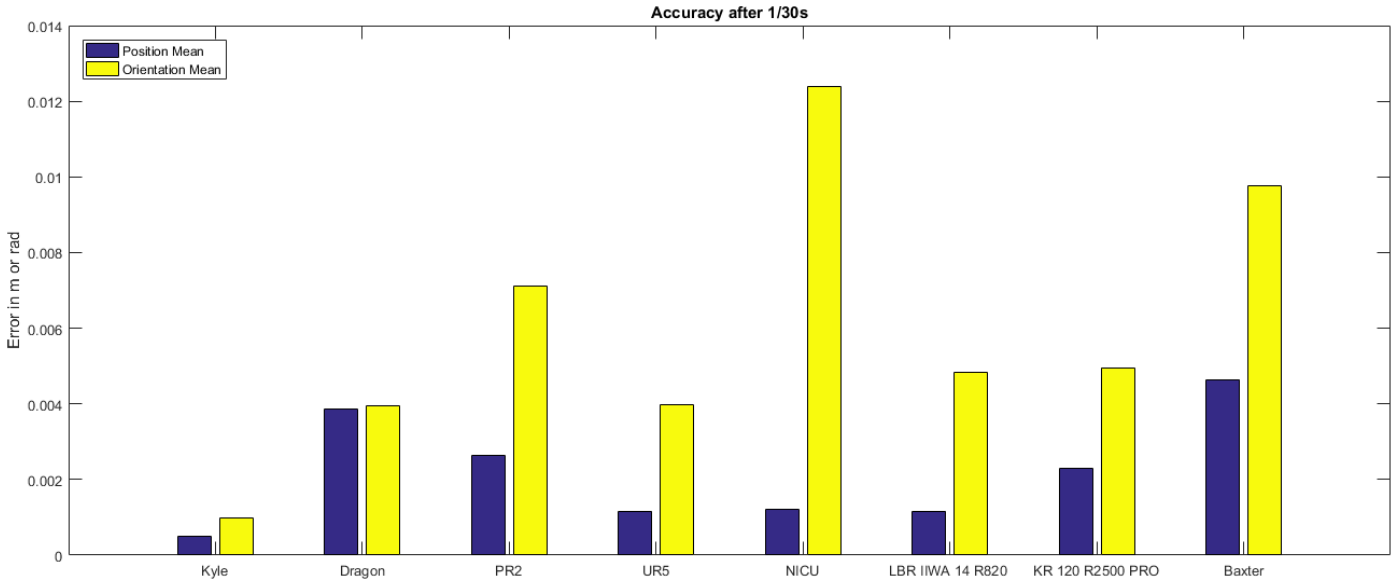


Figure 5.23: Accuracy in Position and Orientation after $\frac{1}{30}$s

error in position can be optimized to very few millimeters while the mean error in orientation shows a slighty higher variation but can reliably achieve errors far below one degree. Obviously, the position error also implicitly depends on the size of the model and the length of its controlled kinematic chain. Again, the best performance in accuracy could be obtained by the Kyle model despite of its comparatively high degree of freedom. However, it is also of particular interest how efficient the algorithm performs in tracking a Cartesian target after a reasonably accurate solution has been found. In this context, Fig. 5.24 demonstrates the pose tracking accuracy while ensuring a constant frame rate of 60Hz. The experiment

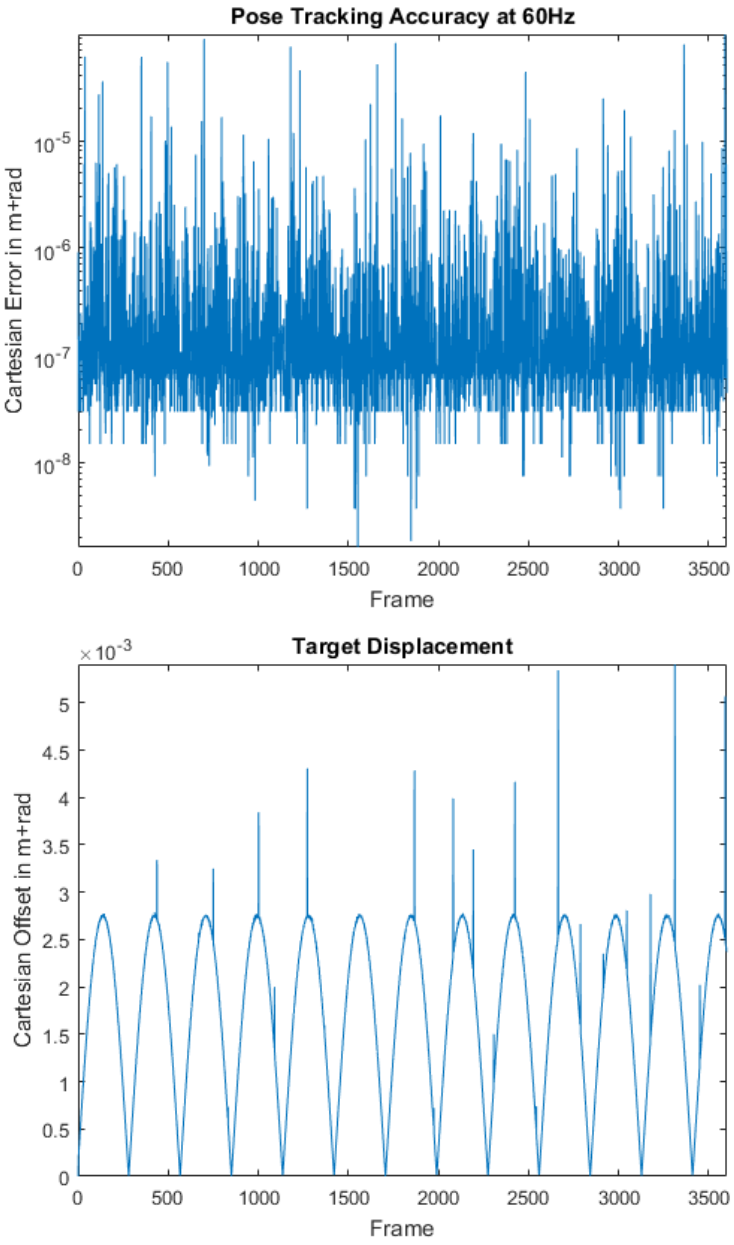


Figure 5.24: Accuracy in Pose Tracking at 60Hz Frame Rate

was conducted using the UR5 robot model where the upper image visualizes the remaining Cartesian error of the end effector with respect to the Cartesian target displacement shown in the lower image. Note that the Cartesian error is calculated as the sum of errors in position and orientation. The noise in the target displacement origins from spikes within the delta time between two consecutive frames which is particularly used in order to create a visually smooth target movement. The results show that it is possible maintain Cartesian errors of 10^{-6} to 10^{-7} for the end effector in average while following a trajectory at interactive frame rates.

5.5.3 Time

A similar study as for the performance in accuracy was also conducted regarding the required computation time. Accordingly, Fig. 5.25 depicts the computation that was neccessary in order to achieve an accuracy of 0.001m and 0.01rad under a pose, position or orientation objective respectively. Clearly, satisfying a pose objective is from a mathematical perspective much more complex than solving an exclusive position or orientation objective and hence requires a higher computational effort. However, the requested solutions for the desired accuracy could still be found within a few milliseconds. Regarding the required amount of generations as previously shown in Fig. 5.22, the computation time strongly correlates to the inherent degree of freedom of the kinematic model. This can particularly be observed by the Dragon model for which only very few generations were needed to converge. Still, those result in a relatively high computation time due to the calculation of the forward kinematics. Further, Fig. 5.26 demonstrates the required computation time during pose tracking at the specified accuracy using the UR5 robot model. In this, the maximum frame rate was unlimited and the results in

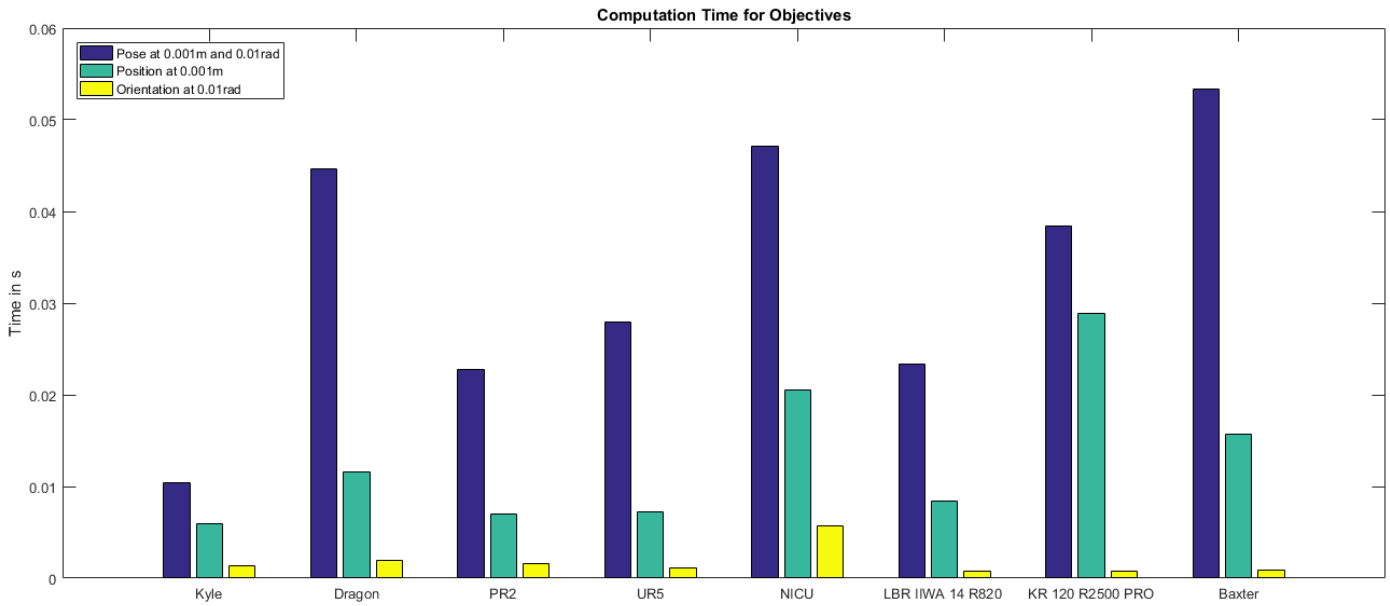


Figure 5.25: Computation Time for Objectives at predefined Accuracy

the upper image show that the algorithm is capable of accurately tracking the given Cartesian target displacement that is depicted within the lower image at an average frame rate of 1000-2000Hz. This is very beneficial since it shows that the algorithm allows to solve multiple inverse kinematics queries simultaneously while still ensuring interactive frame rates. Lastly, Fig. 5.27 illustrates the correlation

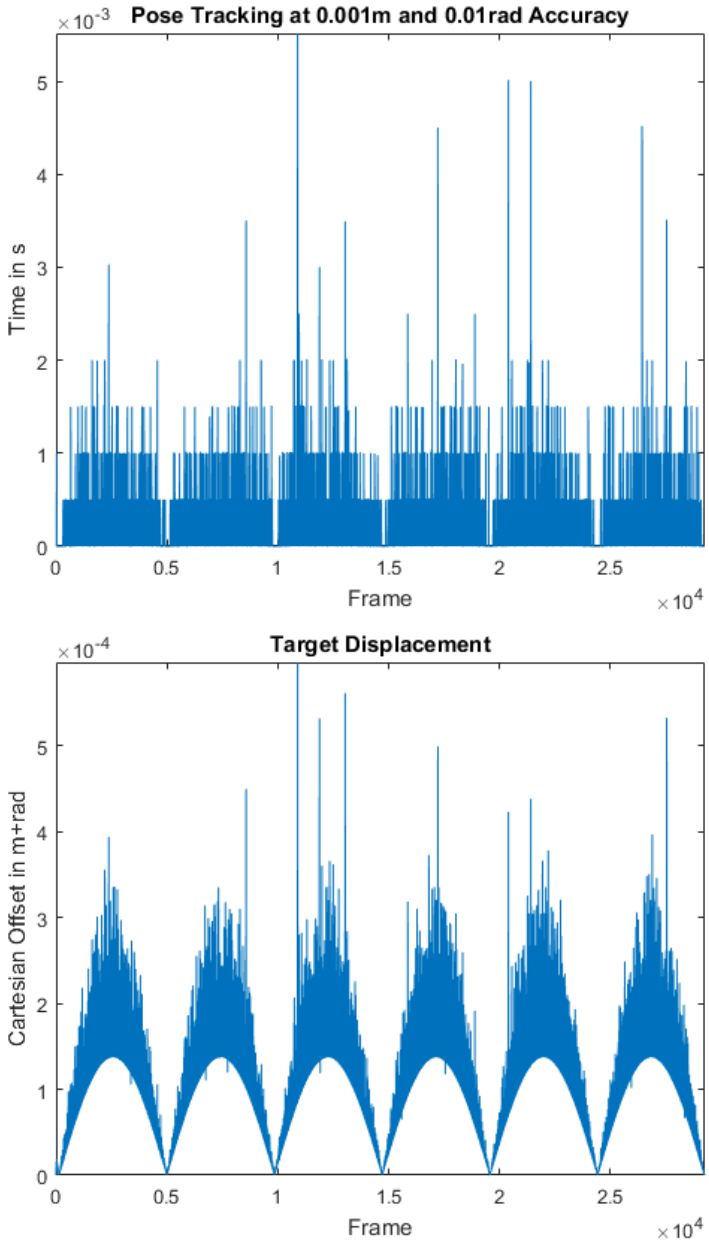


Figure 5.26: Computation Time for Pose Tracking at predefined Accuracy

between the required computation time given a desired pose accuracy regarding different kinematic models. The results suggest an exponential increase in computation time with respect to a logarithmic scale in accuracy. Hence, the performance efficiency between accuracy and time seems to scale linearly with each other.

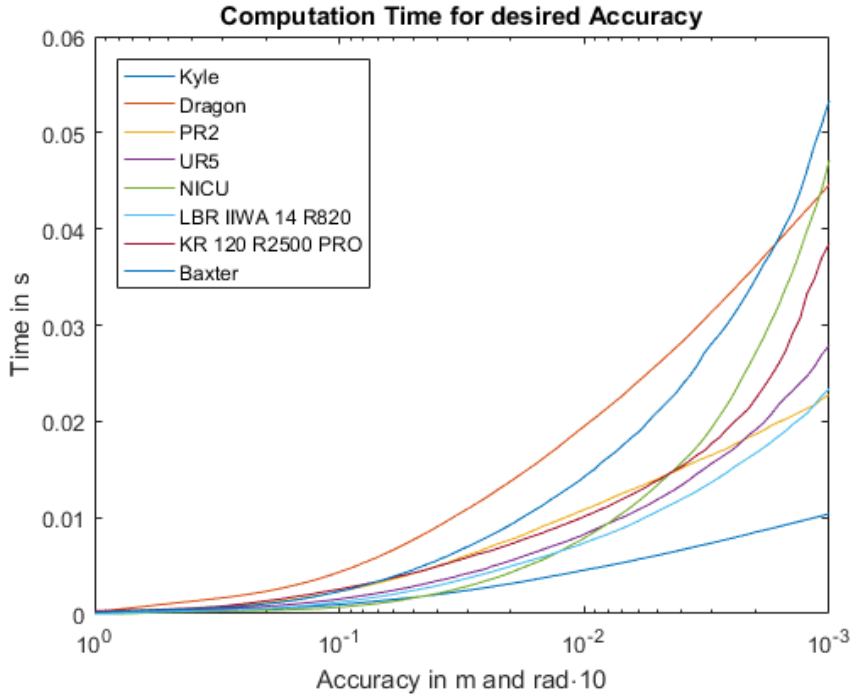


Figure 5.27: Increase in Computation Time with respect to the desired Accuracy

5.5.4 Displacement

An important performance criteria that is especially required for efficient path generation but also to simulate realistic motions in character animation is given by the displacement in joint space relative to Cartesian space. The conducted experiments are conducted on continual trajectories where the end effector is immediately teleported by the joint variable configuration. First, Fig. 5.28 visualizes

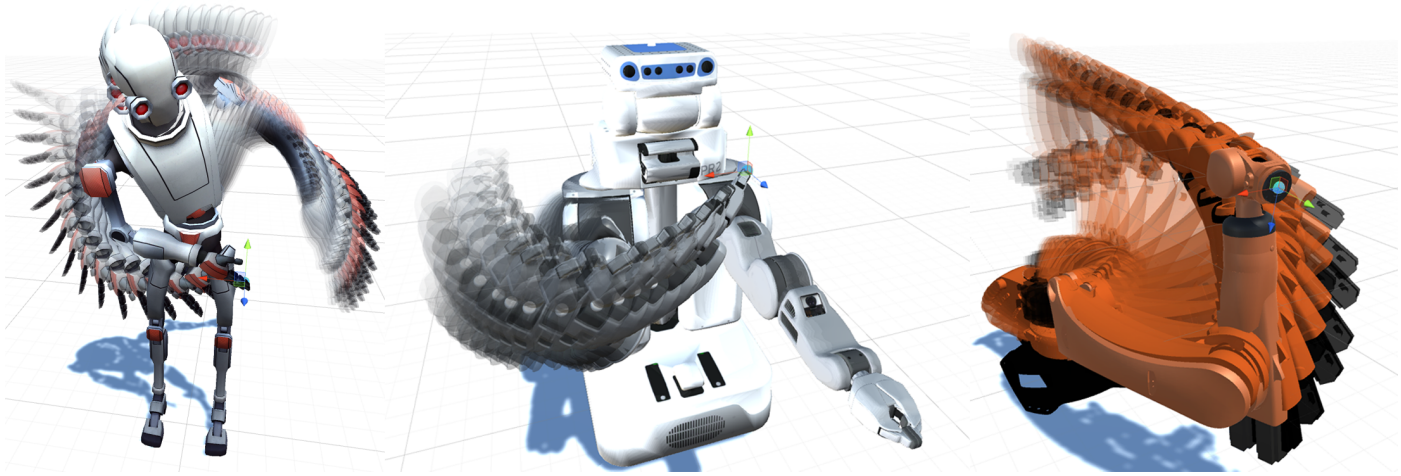


Figure 5.28: Displacement Visualization by following a Cartesian Trajectory

the generated trajectory of the end effector while interactively following a manually displaced target. In this, a very smooth movement can be observed both for a position (left and middle image) as well as for a pose (right image) objective. In more detail, Fig. 5.29 shows the joint displacement measured by the sum of the individual joint value changes along the whole kinematic chain in the upper image relative to a Cartesian target displacement within the lower image. The results reveal a very responsive and shape-resembling joint value change that is even able to accurately react to the noise that is caused by occasional frame spikes which affects the continual target displacement.

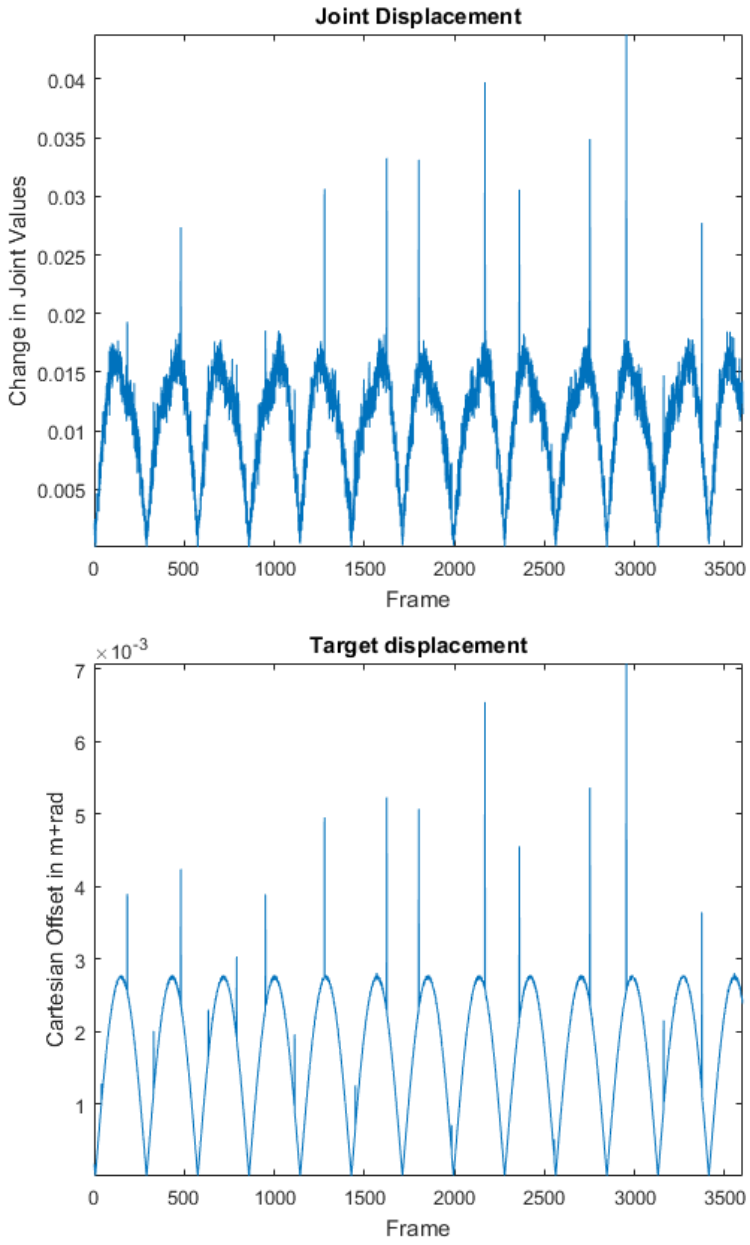


Figure 5.29: Joint Displacement in Pose Tracking at 60Hz Frame Rate

5.5.5 Flexibility

Although the flexibility of the algorithm has already been demonstrated within the previous experiments, this section specifically aims to show the scalability for greatly higher-dimensional degree of freedom, the applicability of different joint types and the robustness if the target can not be reached by any joint variable configuration. Considering the dimensional scalability, Fig. 5.30 depicts three hyper-redundant models with 30, 90 and 180 degrees of freedom that were created by consecutively connected spherical joints. Further, Fig. 5.31 shows the related

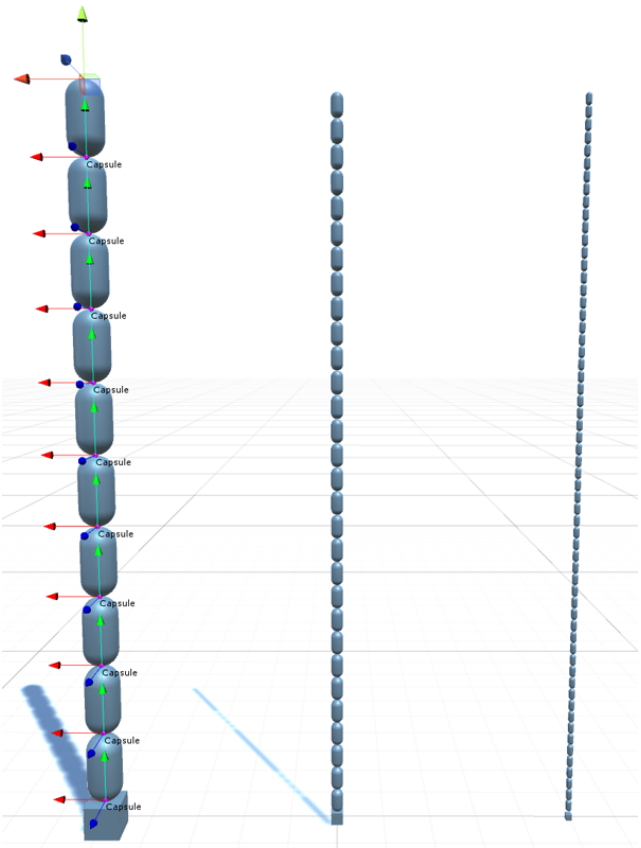


Figure 5.30: Hyper-Redundant Kinematic Models

performance in computation time regarding a rising degree of freedom with respect to a pose objective with 0.001m and 0.01rad accuracy. It can be observed that all solutions were successfully evolved within less than 1s what is especially surprising considering an extremely high degree of freedom. More particularly, many algorithmic approaches which rely on solving the inverse kinematics problem mathematically by deriving an error gradient would usually not be able to obtain a suitable or any solution at all. In terms of the Hybrid Genetic Swarm Algoritm, the increase in computation time seems to mainly depend on the accumulating calculation of the forward kinematics. Regarding the flexibility on different joint types, the corresponding results under equivalent objective specifications are depicted in Fig. 5.32. The results primarily show that the algorithm can efficiently handle a combina-

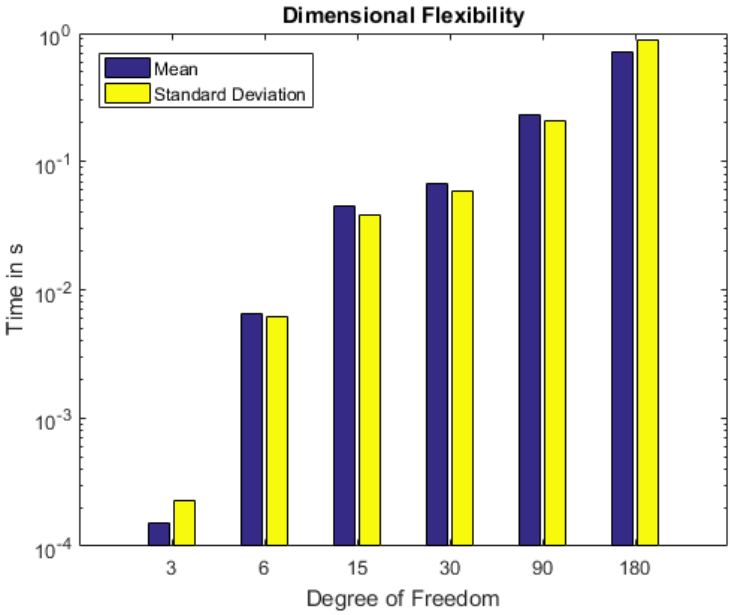


Figure 5.31: Flexibility with rising Degree of Freedom

tion of different joint types where prismatic joints particularly contribute to lower computation times. This can be reasoned regarding that translational movement does not affect the orientation of the end effector and thus implicitly decreases the geometric complexity while rotational movement affects both the position and orientation. Both experiments together clearly demonstrate the universal applicability of the designed algorithm on arbitrary kinematic models. Ultimately, the

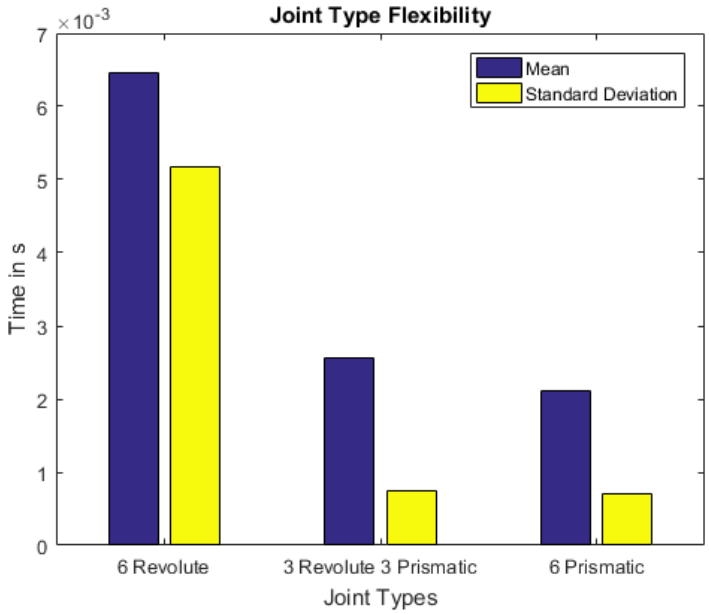


Figure 5.32: Flexibility with different Joint Types

experiment on the depicted models in Fig. 5.33 specifically demonstrates the robustness of the algorithm in case of an unreachable Cartesian target configuration. This particularly shows the advantage in numerical over analytical approaches for inverse kinematics where the latter would not be able to obtain any solution to the given problem.

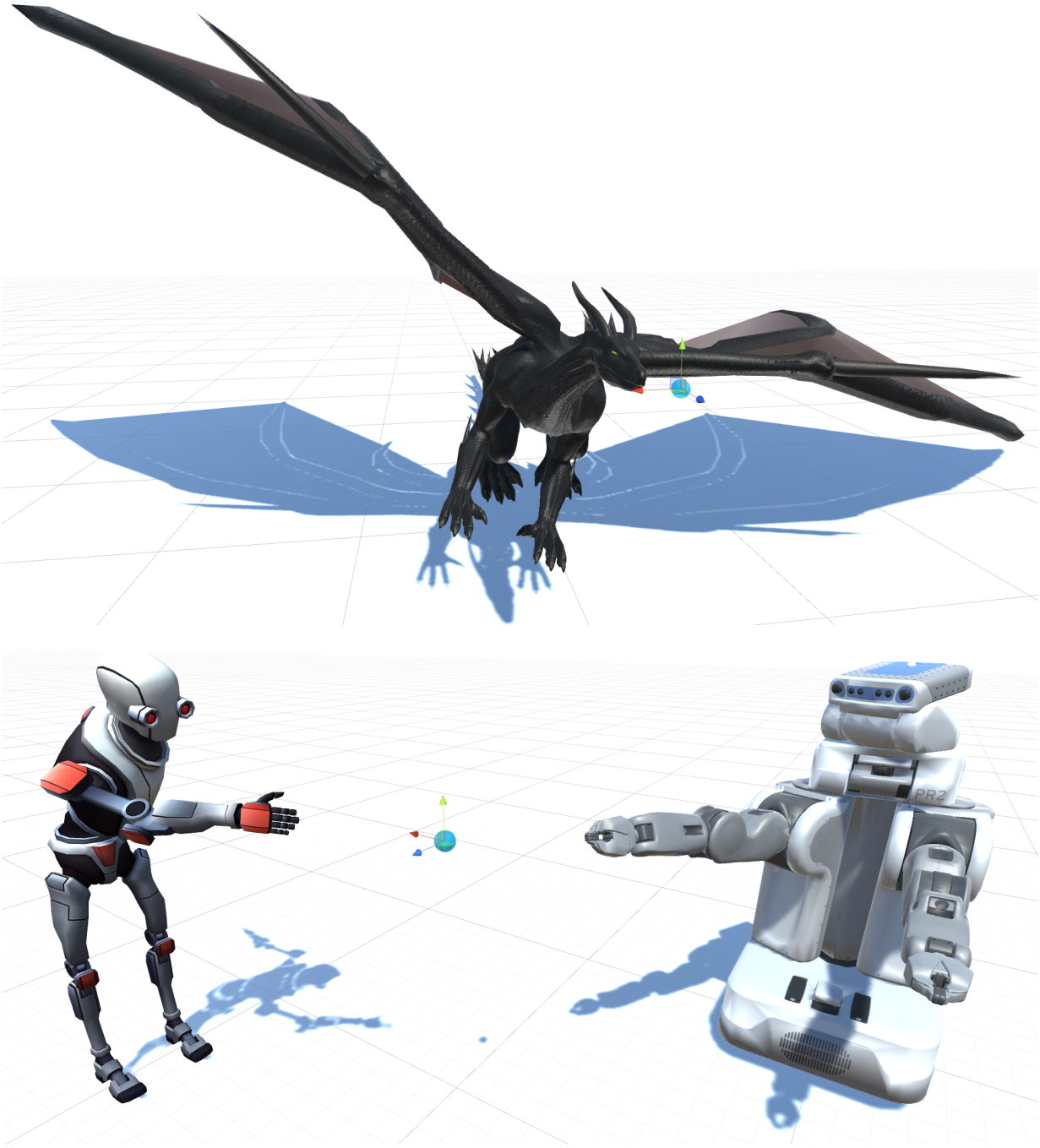


Figure 5.33: Flexibility with unreachable Targets

5.6 Comparative Study

Inverse kinematics is a problem for which many algorithms have been developed that are all tuned for the specific requirements of their applications. Accordingly, it remains difficult to conduct a qualitative comparison that jointly involves all performance aspects. This section mainly aims to provide an approximate classification of the Hybrid Genetic Swarm Algorithm among related approaches.

In this context, Tbl. 5.2 gives a comparison to the widely used Orocos KDL [40] as well as the novel TRAC-IK [7, 8] algorithms which are explicitely designed for robotics and shows the average success rate and computation time on different robot models. While both Orocos KDL and TRAC-IK achieve remarkably low computation times despite of significant differences within their success rates, the HGSA can convince by a constant success rate of 100%. However, the required computation time is considerably higher but still sufficient for many applications.

Manipulator	DoF	Orocos KDL	TRAC-IK	HGSA
Baxter Arm	7	61.07% (2.21ms)	99.17% (0.60ms)	100% (22.75ms)
KUKA LBR iiwa 14 R820	7	37.71% (3.37ms)	99.63% (0.56ms)	100% (23.39ms)
PR2 Arm	7	83.14% (1.37ms)	99.84% (0.59ms)	100% (27.88ms)
UR5	6	35.88% (3.30ms)	99.55% (0.42ms)	100% (53.33ms)

Table 5.2: Comparison between Orocos KDL, TRAC-IK and HGSA

A further comparison in terms of scalability between PASO [15] and HGSA is listed in Tbl. 5.3 and shows the algorithmic efficiency in computation time on hyper-redundant models. In this, HGSA is clearly able to outperform PASO by one or two orders of magnitude under similar kinematic geometry and accuracy.

DoF	PASO	HGSA
30	1.57s	0.066s
90	7.46s	0.233s
180	37.03s	0.717s

Table 5.3: Comparison between PASO and HGSA

Ultimately, Tbl. 5.4 proposes a general overview on the algorithmic efficiency and performance including the numerical approaches for inverse kinematics that were discussed in chapter 3. The values are chosen regarding a full Cartesian pose objective. While the Jacobian and CCD methods were able to achieve the best rates in accuracy and computation time since they always follow the gradient of the steepest functional descent, they typically lack in terms of robustness and scalability and thus work best on certain kinematic geometry. Similar observations in efficiency could be made for the FABRIK approach which uses heuristic geometric computations that do not directly operate in joint space but what is of particular importance for many applications in robotics. Although the algorithm showed good performance in terms of scalability by handling multiple end effectors simultaneously, problems were reported in robustness considering joint constraints as well as different types. In contrast to those, GA and PSO usually resulted in a lower accuracy and required much more computation time that is reasoned by the nature of probabilistic optimization. However, they revealed great capabilities both in robustness and scalability and thus provide a higher flexibility to greatly varying kinematic geometry. ANN approaches that tackle the problem by learning the inverse kinematics function showed highest difficulties in all aspects. Lastly, experiments on the developed HGSA were able to maintain the advantages of GA and PSO in robustness and scalability and showed same accuracy rates but further obtained comparable results in terms of computation time as the Jacobian, CCD and FABRIK methods. This implies that a heuristic hybridization of GA and PSO can achieve a significantly higher efficiency than the single approaches and remains as a generic and universal solution for solving the inverse kinematics problem.

	Jacobian	CCD	FABRIK	GA	PSO	ANN	HGSA
Accuracy (m | rad)	0.00001	0.00001	0.0001	0.001	0.001	0.01	0.001
Time (ms)	$< 1 - 10$	$1 - 100$	$10 - 50$	$50 - 500$	$30 - 600$	-	$10 - 60$
Robustness	Medium	Medium	Low	High	Medium	Low	High
Scalability	– –	–	+	+	++	– –	++

Table 5.4: Algorithmic Comparison on the reported Efficiency and Performance

Chapter 6

Conclusion

"I am among those who think that science has great beauty. A scientist in his laboratory is not only a technician: he is also a child placed before natural phenomena which impress him like a fairy tale." – Marie Curie

This final chapter concludes the work and summarizes results that were attained by this thesis.

First, section 6.1 provides a summary on the content within the previous chapters where particular focus is laid on the algorithmic approach presented in chapter 4 as well as the experimental analysis in chapter 5.

Section 6.2 then gives a list of contributions which evolve from the implementation of the biologically-inspired Hybrid Genetic Swarm Algorithm and its algorithmic improvements and modifications.

Ultimately, section 6.3 finishes this thesis with several suggestions on future work both from a computational as well as scientific perspective.

6.1 Summary

This thesis presented the developed Hybrid Genetic Swarm Algorithm that solves the inverse kinematics problem by means of biologically-inspired optimization techniques and proposes a universal solution that can be applied on arbitrary joint chains.

The work within this thesis was primarily motivated by two different objectives. First, a generic inverse kinematics solution should be designed that achieves reasonably high accuracy while providing real-time capability in order to ensure interactive frame rates. Second, the intent was in creating novel algorithmic and biologically plausible concepts which achieve a higher adaptivity in exploitation and exploration and are applicable to various problems that can be solved by optimization strategies.

Initially, a broad overview on the fundamental knowledge covering the fields of robotics and kinematics was given in chapter 2. This further involved a discussion on the algorithmic methodology regarding analytical and numerical approaches followed by an insight into the area of biologically-inspired artificial intelligence.

Chapter 3 then highlighted the current state of the art with focus on the numerical solutions. Those involved the traditional Jacobian and Cyclic Coordinate Descent methods which iteratively follow the gradient and have shown to be comparatively fast and accurate but lastly suffered from suboptimal extrema and showed only few scalability. The geometric FABRIK approach was specifically designed for character animation and motion tracking and could obtain realistic motion while handling multiple end effectors but provides no direct information about the joint variable configuration since it operates in Cartesian space. The results of artificial neural networks by learning the inverse kinematics function showed least performance where difficulties could be observed in satisfying pose objectives or a higher degree of freedom. Lastly, a critical comparison to genetic algorithms and particle swarm optimization revealed that those typically require a consiberably higher computational cost but offer great opportunities in terms of robustness and dimensional scalability.

The algorithmic approach was then presented in detail within chapter 4. Constructing on the given problem statement and the initial overview on the complete algorithm, the single evolutionary phases and operators were described involving explicit mathematical formalizations. The chosen genetic encoding scheme allows independency to the joint types and to directly incorporate constraints. The design of the fitness function was inspired by natural evolution using randomized weights for multi-objective optimization to model a dynamically changing environment. During recombination of parent chromosomes that are selected according to their rank, the idea was to let offspring dive a little deeper into the direction that heuristically caused improvement within their parents which was denoted as the

evolutionary gradient. The mutation operator was designed to let the population itself determines the required amount of exploitation and exploration in mutation rate and strength by a calculated extinction factor that adaptively reacts to the problem dimensionality and both maintains diversity as well as sensitivity to local extrema. Inspired by the collective swarm dynamics as observed in natural phenomena, the adoption phase aims to let offspring adopt the characteristics of parents and the most successfully performing individual within the population. This concept further allows a dynamic search space exploration and supports the inherent behaviour of organisms that is conducted over lifetime. The niching was then described by immediately removing any parent from the mating pool whose offspring scores a higher fitness value what encourages the population to keep track of multiple local extrema simultaneously and avoid premature convergency. The selection of survivors was chosen to merge all offspring with the elites of the previous generation in order not to lose the current evolutionary progress. More particularly, an additional heuristic exploitation was designed and applied to the elites with intent to significantly increase convergeny. Lastly, a wipe of the whole population was allowed if the population was detected to be entirely stuck where passing the current solution into the reinitialized population allowed to directly continue scoring progress in optimization.

Consequently, an extensive experimental analysis on the algorithm was conducted in chapter 5 regarding its parameters, the individual improvements and modifications, the overall performance as well as a comparison to related approaches. For the required parameters that are given by the population size and the number of elites, the results suggested to choose the former two or three times higher than the latter. The selective study on the algorithmic design showed that the Hybrid Genetic Swarm Algorithm can achieve an efficiency gain by two or three orders of magnitude over simple genetic algorithms. Further experiments on each of the single improvements and modifications showed that removing any of them results in a significant loss in efficiency. The algorithm was then evaluated using various kinematic models of different geometry and regarding five distinct performance criteria in success, accuracy, time, displacement and flexibility. In particular, a constant success rate of 100% on all models over 10.000 randomly generated reachable targets could be obtained. In this, the efficiency in accuracy and computation time was approximately about 0.001m and 0.01rad within 10-60ms. In further observations, tracking a continuously displaced target along a reachable trajectory could maintain a very high accuracy at interactive frame rates with up to 1000-2000Hz. Also, the related joint displacement was very responsive and minimal with respect to the change within the Cartesian target. Experiments on the algorithmic flexibility under a degree of freedom of up 180 and different joint types still reliably obtained solutions and even converged to a numerically optimal configuration if the target was not reachable. Lastly, the comparative study showed that a heuristic hybridization of evolutionary algorithms and collective systems as implemented in the developed Hybrid Genetic Swarm Algorithm can successfully compete with the current state of the art in inverse kinematics.

6.2 Contributions

This thesis provides several contributions both from a scientific as well as technical perspective where this section lists the ones found most relevant.

First, a novel biologically-inspired evolutionary approach for solving the inverse kinematics problem on arbitrary joint chains with high performance in success, accuracy, time, displacement and flexibility was designed.

Second, a hybridization of genetic algorithms with particle swarm optimization was shown to be capable of achieving notably better results than the single approaches.

Third, an extinction factor was introduced with intent to adaptively control the mutation phase in genetic algorithms.

Fourth, an additional integration of a local search technique into biologically-inspired optimization revealed significant efficiency improvements.

Fifth, it was demonstrated how randomization can successfully be applied to the complex mathematical problem of multi-objective optimization.

Sixth, a URDF importer as well as implementations for kinematic joints and inverse kinematics with intuitive and easily adjustable editor components were developed for Unity.

6.3 Future Work

This thesis offers several interesting aspects for further investigations and improvements which reach from purely computational to algorithmic and scientific perspectives.

First, a study on the extinction factor might be considered where particular interest is in the caused loss by the adaptively controlled mutation in contrast to model-specific optimized parameters.

Further, it was observed that >95% of the computation time for optimization was required for solving the forward kinematics equations for fitness calculations. Accordingly, a reimplementation of the algorithm in C++/Python with an efficient parallel GPU or multi-core processing might achieve a significant computational speedup.

Currently, only serial kinematic chains are supported by the algorithmic solution where further work could extend the algorithm to handle multiple end effectors simultaneously.

The evolutionary optimization is primarily driven by the design of the fitness function. In this context, a weighting of joints with respect to their preference and relevance within the kinematic chain is particularly interesting in terms of realistic motion.

A common problem that is especially important in robotics and which directly constructs on the obtained solutions by inverse kinematics is given by path planning where several future work can be done.

Equivalently, the current implementation does not consider self-collision handling for the evolved joint variable configurations for but what is of crucial importance in real world applications.

A further interesting study from a scientific perspective can be performed by integrating a neural learning of previously evolved solutions. Although artificial neural networks showed poor performance for solving the inverse kinematics problem as a whole, improvements can be imagined by feeding learned approximate configurations as guiding individuals into the evolutionary optimization.

Ultimately, the challenge remains in designing more sophisticated solutions that use generically applicable concepts to solve complex problems what will be essential within the next generation of artificial intelligence.

Bibliography

[1] Omar Alejandro Aguilar and Joel Carlos Huegel. *Inverse Kinematics Solution for Robotic Manipulators Using a CUDA-Based Parallel Genetic Algorithm*, volume 7094, pages 490–503. Springer Berlin Heidelberg – Lecture Notes in Computer Science, 2011.

[2] Andreas Aristidou and Joan Lasenby. *Inverse Kinematics: a review of existing techniques and introduction of a new fast iterative solver*. University of Cambridge, 2009.

[3] Andreas Aristidou and Joan Lasenby. *FABRIK: A fast, iterative solver for the Inverse Kinematics problem*, volume 73, pages 243–260. Graphical Models, 2011.

[4] Andreas Aristidou and Joan Lasenby. *Real-time marker prediction and CoR estimation in optical motion capture*, volume 29, pages 7–26. The Visual Computer, 2013.

[5] A. Balestrino, G. De Maria, and L. Sciavicco. Robust control of robotic manipulators. In *Proceedings of the 9th IFAC World Congress*, volume 5, pages 2435–2440, 1984.

[6] Model: Baxter.
`https://github.com/RethinkRobotics/baxter_common`.
Accessed: 26-06-2016.

[7] TRAC-IK.
`https://bitbucket.org/traclabs/trac_ik.git`.
Accessed: 26-06-2016.

[8] Patrick Beeson and Barrett Ames. TRAC-IK: An open-source library for improved solving of generic inverse kinematics. In *Proceedings of the IEEE RAS Humanoids Conference*, Seoul, Korea, November 2015.

[9] Richard Bellman. *Adaptive Control Processes*. Princeton University Press, 1961.

[10] S. R. Buss and J. S. Kim. *Selectively damped least squares for inverse kinematics*, volume 10, pages 37–49. Journal of Graphics Tools, 2005.

[11] Samuel R. Buss. *Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods.* Survey, University of California, 2004.

[12] S. K. Chan and P. D. Lawrence. *General inverse kinematics with the error damped pseudoinverse*, pages 834-839. Proceedings of the IEEE International Conference on Robotics and Automation, 1988.

[13] Darryl Charles, Colin Fyfe, and Daniel Livingstone. *Biologically Inspired Artificial Intelligence for Computer Games.* IGI Publishing, 1st Edition, 2007.

[14] S. Chiaverini, B. Siciliano, and O. Egeland. *Review of damped least-squares inverse kinematics with experiments on an industrial robot manipulator*, volume 2, pages 123-134. IEEE Transactions on Control Systems Technology, 1994.

[15] Thomas Collins and Wei-Min Shen. *PASO: An Integrated, Scalable PSO-based Optimization Framework for Hyper-Redundant Manipulator Path Planning and Inverse Kinematics.* Information Sciences Institute, Technical Report, 2016.

[16] Bassam Daya, Shadi Khawandi, and Mohamed Akoum. *Applying Neural Network Architecture for Inverse Kinematics Problem in Robotics*, volume 3, pages 230–239. Journal of Software Engineering and Applications, 2010.

[17] Jacques Denavit and Richard Scheunemann Hartenberg. *A kinematic notation for lower-pair mechanisms based on matrices*, volume 23, pages 215–221. Transactions in ASME – Journal of Applied Mechanics, 1955.

[18] Jacques Denavit and Richard Scheunemann Hartenberg. *Kinematic synthesis of linkages.* McGraw-Hill Series in Mechanical Engineering, 1965.

[19] Rosen Diankov. *Automated Construction of Robotic Manipulation Programs.* Doctoral Thesis, The Robotics Institute, Carnegie Mellon University, 2010.

[20] Rosen Diankov, Kenji Sato, Hiroaki Yaguchi, Kei Okada, and Masayuki Inaba. *Manipulation Planning for the JSK Kitchen Assistant Robot Using OpenRAVE.* University of Tokyo, 2011.

[21] P. S. Donelan. *Kinematic Singularities of Robot Manipulators*, pages 401–416. Advances in Robot Manipulator, In-Tech, 2010.

[22] Model: Dragon.
`http://tf3dm.com/3d-model/black-dragon-rigged-and-game-ready-92023.html`.
Accessed: 26-06-2016.

[23] Fletcher Dunn and Ian Parberry. *3D Math Primer For Graphics And Game Development.* Jones & Bartlett Learning, 2002.

[24] B. Durmus, H. Temurtas, and A. Gün. *An Inverse Kinematics Solution using Particle Swarm Optimization*, pages 193–197. 6th International Advanced Technologies Symposium, 2011.

[25] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer Verlag Berlin-Heidelberg – Natural Computing Series, 2003.

[26] Unreal Engine.
`https://www.unrealengine.com`.
Accessed: 26-06-2016.

[27] Unreal Engine - FABRIK.
`https://docs.unrealengine.com/latest/INT/Engine/Animation/NodeReference/Fabrik/`.
Accessed: 26-06-2016.

[28] Yin Feng, Wang Yao-nan, and Yang Yi-min. *Inverse Kinematics Solution for Robot Manipulator based on Neural Network under Joint Subspace*, volume 7, pages 459–472. International Journal of Computers Communications and Control, 2012.

[29] Dario Floreano and Claudio Mattiussi. *Bio-Inspired Artificial Intelligence – Theories, Methods, and Technologies*. MIT Press, 2008.

[30] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution*. Wiley, 1966.

[31] ROS Framework.
`http://www.ros.org`.
Accessed: 26-06-2016.

[32] John Q. Gan, Eimei Oyama, Eric M. Rosales, and Huosheng Hu. *A complete analytical solution to the inverse kinematics of the Pioneer 2 robotic arm*, volume 23, pages 123–129. Robotica, 2005.

[33] H. Hanafusa and Y. Nakamura. Inverse kinematics solutions with singularity robustness for robot manipulator control. In *Journal of Dynamic Systems, Measurement, and Control*, volume 108, pages 163–171, 1986.

[34] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath, and Brian Kingsbury. *Deep Neural Networks for Acoustic Modeling in Speech Recognition*, volume 29, pages 82–97. Signal Processing Magazine, 2012.

[35] John H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[36] Panchanand Jha and BB Biswal. *A Neural Network Approach for Inverse Kinematic of a SCARA Manipulator*, volume 3, pages 52–61. International Journal of Robotics and Automation, 2014.

[37] D. Karger and C. Stein. *A new approach to the minimum cut problem*, volume 43. Journal of the ACM, 1996.

[38] J. Kennedy and R. Eberhart. *Particle Swarm Optimization*, volume 4, pages 1942–1948. Proceedings of the IEEE International Conference on Neural Networks, 1995.

[39] Ben Kenwright. *Inverse Kinematics - Cyclic Coordinate Descent (CCD)*, volume 16, pages 177-217. Journal of Graphics Tools, 2012.

[40] Orocos Kinematics and Dynamics.
`http://www.orocos.org`.
Accessed: 26-06-2016.

[41] Donald E. Knuth. *The Art of Computer Programming, Volumes 1-4, 3rd Edition*. Addison Wesley, 2011.

[42] Rasit Köker. *A genetic algorithm approach to a neural-network-based inverse kinematics solution of robotic manipulators based on error minimization*, volume 222, pages 528–543. Information Sciences – New Trends in Ambient Intelligence and Bio-inspired Systems, 2013.

[43] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. *The fantastic combinations of John Conways new solitaire game "life"*, volume 223, pages 120–123. Scientific American, 1970.

[44] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. *On cellular automata, self-reproduction, the Garden of Eden and the game "life"*, volume 224, pages 112–117. Scientific American, 1971.

[45] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. *Imagenet classification with deep convolutional neural networks*, volume 25, pages 1106–1114. Advances in Neural Information Processing Systems, 2012.

[46] Serdar Kucuk and Zafer Bingul. *Industrial Robotics: Theory, Modelling and Control*. pro literatur Verlag, 2007.

[47] Model: Space Robot Kyle.
`https://www.assetstore.unity3d.com/en/#!/content/4696`.
Accessed: 26-06-2016.

[48] Jeff Lander. *Making kine more flexible*, volume 5, pages 15-22. Game Developer, 1998.

[49] S. Lawrence, C. L. Giles, Ah Chung Tsoi, and A. D. Back. *Face Recognition: A Convolutional Neural-Network Approach*, volume 8, pages 98–113. Signal Processing Magazine, 1997.

[50] Jeffery J. Leader. *Numerical Analysis and Scientific Computation, 1st Edition.* Pearson, 2004.

[51] Eric Lengyel. *Mathematics for 3D Game Programming and Computer Graphics, 3rd Edition.* Cengage Learning PTR, 2011.

[52] D. G. Lowe and M. Muja. *Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration*, pages 331–340. INSTICC Press, 2009.

[53] Unity3D Inverse Kinematics Manual.
`http://docs.unity3d.com/Manual/InverseKinematics.html`.
Accessed: 26-06-2016.

[54] Unreal Engine Inverse Kinematics Manual.
`https://docs.unrealengine.com/latest/INT/Engine/Animation/IKSetups/`.
Accessed: 26-06-2016.

[55] R. V. Mayorga, A. K. C. Wong, and N. Milano. *A fast procedure for manipulator inverse kinematics evaluation and pseudoinverse robustness*, volume 22, pages 790–798. IEEE Transactions on Systems, Man, and Cybernetics, 1992.

[56] J. M. McCarthy and G. S. Soh. *Geometric Design of Linkages, 2nd Edition.* Springer, 2010.

[57] Michael Meredith and Steve Maddock. *Real-Time Inverse Kinematics: The Return of the Jacobian.* Department of Computer Science, University of Sheffield, Technical Report, 2004.

[58] Root Motion.
`http://root-motion.com`.
Accessed: 26-06-2016.

[59] Ramakrishnan Mukunda. *Advanced Methods in Computer Graphics.* Springer-Verlag London, 2012.

[60] R. Müller-Cajar and R. Mukundan. *Triangulation: A new algorithm for Inverse Kinematics*, pages 181–186. Proceedings of Image and Vision Computing New Zealand, 2007.

[61] Richard M. Murray, Zexiang Li, and S. Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation.* CRC Press, 1994.

[62] Wolfgang Notling. *Klassische Mechanik – Grundkurs Theoretische Physik 1.* Springer-Verlag Berlin Heidelberg, 2013.

[63] J. K. Parker, A. R. Khoogar, and D. E. Goldberg. *Inverse kinematics of redundant robots using genetic algorithms*, volume 1, pages 271–276. IEEE International Conference on Robotics and Automation, 1989.

[64] E. Pennestri, M. Cavacece, and L. Vita. *On the Computation of Degrees-of-Freedom: A Didactic Perspective.* Proceedings of the International Design Engineering Technical Conference, 2005.

[65] Rolf Pfeifer, Max Lungarella, and Fumiya Iida. *Self-Organization, Embodiment, and Biologically Inspired Robotics*, volume 318, pages 1088–1093. Science, 2007.

[66] Model: PR2.
`http://wiki.ros.org/pr2_description`.
Accessed: 26-06-2016.

[67] Model: KR120 R2500 PRO.
`https://github.com/ros-industrial/kuka_experimental/tree/indigo-devel/kuka_kr120_support`.
Accessed: 26-06-2016.

[68] Model: LBR IIWA 14 R820.
`https://github.com/ros-industrial/kuka_experimental/tree/indigo-devel/kuka_lbr_iiwa_support`.
Accessed: 26-06-2016.

[69] M. O. Rabin. *Probabilistic Algorithms in Algorithms and Complexity*, pages 21–39. Academic Press, 1976.

[70] I. Rechenberg. *Cybernetic Solution Path of an Experimental Problem. Royal Aircraft Establishment.* Royal Aircraft Establishment, Ministry of Aviation, Farnborough Hants, UK, 1965.

[71] A. Chennakesava Reddy. *Difference Between Denavit-Hartenberg (D-H) Classical and Modified Conventions for Forward Kinematics of Robots with Case Study.* International Conference on Advanced Materials and Manufacturing Technologies, 2014.

[72] Claudio Melchiorri Riccardo Falconi, Raffaele Grandi. *Inverse Kinematics of Serial Manipulators in Cluttered Environments using a new Paradigm of Particle Swarm Optimization*, volume 19, pages 8475–8480. The International Federation of Automatic Control, 2014.

[73] Nizar Rokbani and Adel M. Alimi. *IK-PSO, PSO Inverse Kinematics Solver with Application to Biped Gait Generation*, volume 58, pages 33–39. International Journal of Computer Applications, 2012.

[74] Nizar Rokbani and Adel M. Alimi. *Inverse Kinematics Using Particle Swarm Optimization, A Statistical Analysis*, volume 64, pages 1602–1611. International Conference on Design and Manufacturing, 2013.

[75] Joanna Svantesson and Jonas Bornold. *A Real-Time Adaptation of Inverse Kinematics for Motion Capture*. Master Thesis, University of Gothenburg, 2015.

[76] Saleh Tabandeh, Christopher M. Clark, and William W. Melek. *A Genetic Algorithm Approach to solve for Multiple Solutions of Inverse Kinematics using Adaptive Niching and Clustering*, pages 1815–1822. IEEE Congress on Evolutionary Computation, 2006.

[77] Saleh Tabandeh, Christopher M. Clark, and William W. Melek. *An adaptive niching genetic algorithm approach for generating multiple solutions of serial manipulator inverse kinematics with applications to modular robots*, volume 28, pages 493–507. Robotica, 2010.

[78] Sebastian Thrun, Dieter Fox, Wolfram Burgard, and Frank Dellart. *Robust Monte Carlo Localization for Mobile Robots*, volume 128, pages 99–141. Artificial Intelligence, 2001.

[79] Deepak Tolani, Ambarish Goswami, and Norman I. Badler. *Real-Time Inverse Kinematics Techniques for Anthropomorphic Limbs*, volume 62, pages 353–388. Graphical Models, 2000.

[80] A. M. Turing. *Computing Machinery and Intelligence*, volume 59, pages 433–460. Mind, 1950.

[81] Unity3D.
`http://www.unity3d.com`.
Accessed: 26-06-2016.

[82] Model: UR5.
`http://wiki.ros.org/ur5_description`.
Accessed: 26-06-2016.

[83] Model: UR5. Provided by WTM Group, University of Hamburg.

[84] Carla Elena González Uzcátegui. *A Memetic Approach to the Inverse Kinematics Problem for Robotic Applications*. Doctoral Thesis, Carlos III University of Madrid, 2014.

[85] John v. Neumann. *The General and Logical Theory of Automata*, volume 5, pages 288–328. Pergamon Press, 1951.

[86] Nikolaus Vahrenkamp, Tamin Asfour, and Rüdiger Dillmann. Efficient inverse kinematics computation based on reachability analysis. *International Journal of Humanoid Robotics*, 9:2, 2012.

[87] C. W. Wampler. *Manipulator inverse kinematic solutions based on vector formulations and damped least squares methods*, volume 16, pages 93–101. IEEE Transactions on Systems, Man, and Cybernetics, 1986.

[88] Li-Chun Tommy Wang and Chih Cheng Chen. *A combined optimization method for solving the inverse kinematics problems of mechanical manipulators*, volume 7, pages 489–499. IEEE Transactions on Robotics and Automation, 1991.

[89] Tiago Oliveira Weber and Wilhelmus A. M. Van Noije. *Design of Analog Integrated Circuits Using Simulated Annealing/Quenching with Crossovers and Particle Swarm Optimization*. InTech, 2012.

[90] Xiulan Wen, Danghong Sheng, and Jiacai Huang. *A Hybrid Particle Swarm Optimization for Manipulator Inverse Kinematics Control*, volume 5226, pages 784–791. International Conference on Intelligent Computing, 2008.

[91] D. E. Whitney. *Resolved motion rate control of manipulators and human prostheses*, volume 10, pages 47–53. IEEE Transactions on Man-Machine Systems, 1969.

[92] J.H. Wilkinson. *The Algebraic Eigenvalue Problem*. Clarendon Press, 1965.

[93] W. A. Wolovich and H. Ellio. A computational technique for inverse kinematics. In *Proceedings of the 23rd IEEE Conference on Decision and Control*, pages 1359–1363, 1984.

[94] David H. Wolpert and William G. Macready. *No Free Lunch Theorems for Optimization*, volume 1, pages 67–82. IEEE Transactions on Evolutionary Computation, 1997.

Erklärung der Urheberschaft

Ich versichere an Eides statt, dass ich die vorliegende Master Thesis im Studiengang Informatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Insbesondere wurden dabei keine nicht im Quellenverzeichnis benannten Internet-Quellen verwendet. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Ort, Datum Unterschrift

Erklärung zur Veröffentlichung

Ich erkläre hiermit mein Einverständnis zur Einstellung dieser Master Thesis in den Bestand der Bibliothek.

Ort, Datum Unterschrift