TEL AVIV UNIVERSITY
THE IBY AND ALADAR FLEISCHMAN
FACULTY OF ENGINEERING
אוניברסיטת תל-אביב
הפקולטה להנדסה
על שם איבי ואלדר פליישמן

Optimal Configuration of a Robotic Manipulator to Perform a Specific Task in a Cluttered Environment

A thesis
submitted in partial fulfilment
of the requirements for the Degree
of
Master of Science

by

Inbar Meir

This research was carried out at
in the School of Mechanical Engineering
Faculty of Engineering
under the supervision of
Dr. Avishai Sintov, School of Mechanical Engineering
and
Prof. Avital Bechar, Volcani Center

Tel Aviv University

August 2022

Abstract

Robotic arms are the foundation of modern automation for manufacturing. They accommodate production lines and perform the majority of tasks such as assembly, machining, painting, welding and packaging. However, these highly capable robots are usually degraded to simple repetitive tasks such as pick-and-place or welding along the same course. On the other hand, designing an optimal robot for one specific task consumes large resources of engineering time and costs. Moreover, common methods search for collision free paths for robotic arms. However, these are unlikely to be found in cluttered environments where objects must be cleared in order to reach the goal. Furthermore, robotic manipulators are usually interact with the environment solely using their end-effector.

This work seeks for the near-optimal robot configuration to perform a specified task based on human demonstration. The proposed method searches for the robot design variables and the robot placement in the world. An optimal robot is the one that incorporates the minimal DOF and provides best accuracy during task execution. In addition, the optimal design is given with a required path to complete the task. The path is composed using a meta-heuristic method in order to find the joint values to perform the task. This approach takes into consideration the entire robot arm (joint and links), to perform tasks in cluttered environments or to avoid obstacles. The proposed method can also be used to plan a robot path along a human demonstration for existing robots. We provide a comparative analysis to identify the most suitable algorithm to solve this optimization problem. Different known methods, such as Artificial Bee Colony, Genetic Algorithm and Simulated Annealing, are tested and compared. Furthermore and to overcome the highly non-linear and non-continuous search space of the problem, a new algorithm is proposed termed *Robot Arm - Particle Swarm Optimization* (RA-PSO). Moreover, we define a new evaluation index *Normalized Computation Effort Index* (NCEI) that combine the convergence iteration and valid particles. The new method can find the same robot design of the standard method with lower NCEI.

To test and establish our method, we use three diffident test cases. The first is a classic pick and place of an object from a conveyor while avoiding an obstacle. The second scenario is of a robot performing a welding task and requires to track position and orientation of the human demonstrator. The last scenario is a robot moving inside a palm tree canopy in order to perform the thinning task. The palm tree canopy represents a cluttered environment that a collision free path will not be found by the classic path planing methods. In brief, the results show improvement in the NCEI of 90% ,69%, 53% and 44% in average for all scenarios for $DOF = 3, 4, 5, 6$, respectively.

Table of Contents

Acknowledgments

I would like to thank my advisors Prof. Avital Bechar and Dr. Avishai Sintov for their guidance, advice, listening, patience, creativity, support and for granting me the opportunity to learn and perform this research with them. Second, I would like to thank the Ministry of Innovation, Science and Technology for granting me the Golda Meir scholarship and allowed me to focus on this research.

I want to thank the farmer Gazy Moshe who allowed us to perform experiments in his plantation and equipped us with the necessary equipment. Dr. Yuval Chohen for sharing his knowledge and teaching me a lot about palm tree growth and biology, and to Iftach Afgin for helping me work and build all the tools necessary for this work. Moreover, I want to thank all the amazing people I met, worked and consulted in the Institute of Agricultural Engineering. I was amazed by your work and learned allot on this important research field.

I want to thank my laboratory friends, from both Tel Aviv and Volceni center, for all your advice and insights along the way. In addition, I would like to acknowledge my family for supporting, encouraging and believing in me along this journey. Finally, I want to thank my husband Dekel Meir, for listening to me speaking and thinking out loud on this research all this time, and for helping me build all the assistant tools for this research. I couldn't have done this without you.

List of Figures

List of Tables

List of Symbols

symbol	meaning
n	Number of DOF for a manipulator.
m	Number of tracked markers on the arm.
P_i	A path of the i_{th} marker relative to the shoulder.
P	Set of P_m
$p_{i,t}$	The spatial coordinates of the i_{th} marker at time t.
u_t	The Euler angles of the hand or tool at time t.
A_i	Transformation matrix defining the position and orientation of joint i relative to frame $i-1$.
q_i	Joint angle values.
q	Vector of joint values.
Φ	Feature vector defining the design of a robot.
σ	One-dimensional representation of the arm at configuration q.
σ_i	Associates $p_{i,t}$ to the closest point on the robot.
$b_\Phi(q)$	Outputs the Euler angles of the EE for robot Φ at configuration q.
δ_m	User define value to create a list of points on the tested robot.
$g_{\Phi(t,q)}$	The weighted root-mean-square error (wRMSE) between points $p_{1,t}, ..., p_{m,t}$ and the robot.
w_i	User-defined weight values that prioritize the position importance of the robot segments relative to the recorded path.
$G_{\Phi,t}$	Temporal robot fitness at time t for a robot.
$f(\Phi)$	Path fitness is the mean of the temporal finesses for all time frames.
$E(\Phi)$	The mean distance along the robot length and at all time frames.
$L(\Phi)$	Robot length.
Φ^*	Robot path fitness.
λ_f	User definition weighting values for $f(\Phi)$.
λ_E	User definition weighting values for $E(\Phi)$.
Ω_f	The allowed robot design search space.
$f^*(\Phi)$	Alternative tested approach for robot fitness.
C_f	The allowed joint search space.
q_{min}	Minimum actuators angle limits.
q_{max}	Maximum actuators angle limits.
a_{min}	Pre-defined user variables to limit the static values for the DH parameter for link length at x axis.
a_{max}	Pre-defined user variables to limit the static values for the DH parameter for link length at x axis.

(Continued in the next page)

symbol	meaning
d_{min}	Pre-defined user variables to limit the static values for the DH parameter for link length at z axis.
d_{max}	Pre-defined user variables to limit the static values for the DH parameter for link length at z axis.
L_{min}	User-defined values for minimum allowed robot length.
L_{max}	User-defined values for maximum allowed robot length.
N	Number of particles.
M	Iteration Number.
c_{min}	User definition for adapter regain in RA-PSO.
c_{max}	User definition for adapter regain in RA-PSO.
D	Update frequency of the angular variables.

1 Introduction

While robotic arms are highly capable with multiple degrees-of-freedom (DOF), a robot in manufacturing line is usually degraded to solely perform one simple repetitive task [1–4]. These robotic arms are commonly standard off-the-shelve hardware which are redundant for the task at hand and, therefore, highly expensive having direct impact on the final product cost. Hence, many highly capable and expensive robots are purchased and integrated just to perform simple tasks. While simpler and less expensive robots can usually perform the same tasks, they are rarely available as an off-the-shelve product [5]. On the other hand, designing designated robots for each specific task requires complex and careful work by professional engineers. The process consumes a considerable amount of engineering time and may add extra cost to the final product. Hence, an optimally designed kinematic structure is expected to improve performance and reduce costs.

While integrating a new robot, a path to be repeated over and over is usually pre-recorded by a technician while considering robot coordinates [6] or in simulation. The recording is acquired through the teach pendant which provides non-intuitive and non-natural control abilities. Consequently, the resulted path may be non-optimal to the task nor to the environment. Another way to teach a robot a trajectory is to use a path planner. However, common methods search for collision free paths for robotic arms [7–10]. These are unlikely to be found in cluttered environments where objects must be cleared in order to reach the goal. Hence, a future solution should enable the robot to push obstacles, when a collision-free path is not available. Robotic manipulators are usually designed for interaction with the environment solely using their end-effector [11]. Pushing or interacting with obstacles using their body links can be much more efficient in cluttered environments.

For example and as a test case, date thinning of palm trees with a robotic arm while moving in a cluttered environment is investigated. The thinning is required in order to improve fruit quality (size and taste). Currently, the thinning action is the most labor intensive task in the crop. Date trees are very tall (up to heights of 18 meters) and require a crane to reach a date cluster. Such an operation requires immense resources and efforts by the farmer. Moreover, tree tops are extremely cluttered with branches that must be pushed away in order to reach the date clusters. We developed a dynamic model for the Date Majhul tree leaf in order to learn interaction forces with leaves.

In this work, we propose a novel concept of task-oriented robot design based on expert demon-

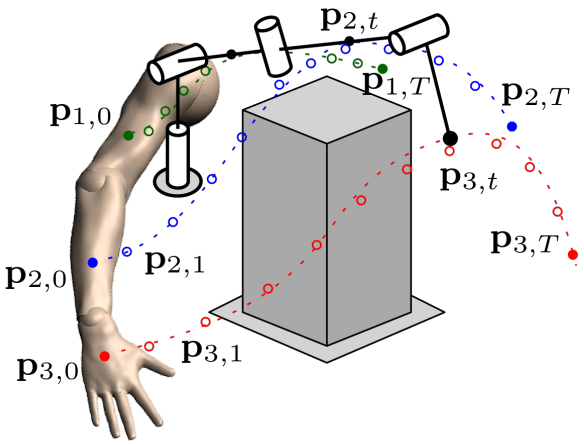


Figure 1.1: Illustration of a robot kinematics evaluated on a recorded task. In this example, the robot end-effector must follow a recorded path of the human hand (circular red markers) while its arm should follow the human arm markers (circular blue and green markers) to avoid obstacles. The proposed optimization method searches for the robotic arm kinematics with the minimum degrees-of-freedom that can best track the recorded task. The dashed curves are the paths of the robotic arm that best track the recorded human ones.

stration able to execute path in cluster environments. We observe a human expert performing a task and assume near-optimal execution. The whole arm motion of the expert is recorded by tracking paths of markers from shoulder to hand (or grasped tool). Then, we formulate an optimization problem in which we search for the minimal DOF robotic arm design that can accurately track the recorded task. In order to be able to avoid or push the obstacles as the expert did, tracking includes the End-Effector (EE) of the robot as well as its entire kinematic chain as seen in Figure 1.1. The optimization framework searches for the Denavit-Hartenberg (DH) parameters [12] that define a robot kinematics.

We compose an optimization problem to optimize the robot joint and link design variables and the placement of the robot in space in order to perform the task. We compare between common meta-heuristic approaches to find the suitable algorithm for the optimization problem. Finally we propose the *Robot Arm - Particle Swarm Optimization* (RA-PSO) algorithm to efficiently solve the design problem. RA-PSO is a modified version of the known *Particle Swarm Optimization* (PSO) method [13] and is particularly aimed to optimize robotic arms based on recorded paths. RA-PSO has modified rules for efficient exploration of the valid search space. The search space in this problem is highly non-linear and non-continuous. Since we optimize all joint and link design variables and robot placement, direct exploration over the entire search space may have trouble converging to the best robot configuration in limited hardware and time constraints. Moreover, we define a new index termed *Normalized Computation Effort Index* (NCEI) that combines the computational effort with the results of the optimization problem. Our approach distributes the search over different design variables to enable a more efficient computation and low NCEI.

The method does not only provide an optimal robot design but also defines the joint path to complete the task. Hence, no additional motion planning is required. Consequently, the proposed process provides the optimal design, robot placement and motion of a robot while reducing the need for engineering work. Using this approach, a novice user can simply record a path and acquire both a robot design and a suitable motion even in static cluttered environments. When the system and task environment are much larger than human capabilities, a scaled mock-up can be used with the proposed approach to compute an optimal robot and later be scaled-up. This approach could also motivate the use of standard and modular robotic hardware, as proposed in [14], such that the novice user can easily assemble and operate custom arms based on the output of the optimization. The approach can also teach a standard off-the-shelf manipulator how to move along a recorded demonstration path.

1.1 Objectives

This overall objective of this work is to find a near-optimal robot configuration to perform a recorded path demonstrated by a human expert. Moreover, we aim to solve a multi-point Inverse Kinematics for a robot configuration such that all robot links can interact with the environment.

A specific aim of this work is to find the optimal design of a robotic arm for date thinning in palm tress. The palm tree can be considered as a cluttered environment such that the arm is required to push the leaves in order to reach the dates.

In addition, we search for the best algorithm to solve the robot configuration optimization problem. Such algorithm is required to find the optimal robot configuration that can track the recorded path. Due to the large computation effort needed in order to solve the optimization problem in continues space, we seek to lower the computation effort while maintaining the performance of known algorithms.

2 Related Work

2.1 Optimal design of robotic arms

2.1.1 Optimal design for increasing dexterity methods

Many research studies have addressed the kinematic optimization of robotic arms while focusing mostly on increasing dexterity and workspace volume. In early work, Vijaykumar et al. [15] focused on general purpose manipulators and maximised their dexterous workspace by simply defining their kinematic geometry. They have shown that for an optimal geometry, singular positions can be completely excluded with small reductions of the joint motion ranges. Patel and Sobh [16] used Grashof's criterion to define the mobility of a four-link closed kinematic chain mechanism in relation to its link lengths. The work used the Grashof's criterion for determining the optimal link lengths of a three-link manipulator, in order to achieve full dexterity at desired regions of the workspace. The work showed dexterity index plots of the workspace generated by applying Grashof's criterion in task placement and trajectory planning.

Recent work [17] used the DH [12] representation to define the design variables of an eight degrees-of-freedom (DOF) upper-limb rehabilitation exoskeleton. A constrained multi-objective optimization problem was formulated utilizing the Genetic Algorithm (GA) to search for the design variables that provide minimum size and maximum dexterity. The kinematic design optimization of an anthropomorphic robot using GA was proposed to maximize the multi-fingered precision grasping capability of the robot hand [18]. Ceccarelli and Lanni [19] presented a multi-objective optimization problem by using the workspace volume and robot dimensions as objective functions. Given workspace limits as constraints for a 3 DOF manipulator with revolute joints, they solved an optimization problem using SQP. Gosselin and Guillot [20] synthesized manipulators such that their workspace is as close as possible to being equal to a prescribed working area. The workspace boundary was described as a list of circular arcs or segments using [21].

Other researchers optimized the robot design variables focusing on the dynamic performance of the manipulator. Ma and Angeles [22] optimized the design of a manipulator under a dynamic isotropy condition. They chose the kinematic and inertial parameters of the manipulator to be the design variable. The dynamic conditioning index is based on the dynamic isotropy concept [23]. However and for some manipulators, an isotropic generalized inertia matrix may be unfeasible.

Jun and Zheng [24] presented a new method for optimal design that takes the dynamic characteristics of parallel manipulators into account. The method was tested on a planar parallel manipulator built with four arms, each with three DOF. In this method, the kinematic and dynamic performance indices are investigated and the corresponding atlases are represented graphically in the established design space. Based on these atlases, non-dimensional geometrical parameters are determined. Then, the optimal dimensional parameters are achieved based on the optimal non-dimensional result of the atlases. Abdel-Malek and Yu [25] suggested a numerical method for the placement of robot manipulators based on maximizing the dexterity at specified target points.

2.1.2 Optimal design for a specific task

While the above methods address enhancing capabilities for general tasks, others have addressed a task-specific optimization problem. Park et al. [26] presented a task-oriented design method for robot kinematics based on the Grid Method. Such method is widely used in the finite difference method or numerical analysis of heat transfer and fluid flow. However, this work uses only perpendicular joints for the robot configuration, which significantly decreases the search space. Yang and Chen [27] searched for modular reconfigurable robot systems with minimal DOF for a specific task while using an evolutionary algorithm (EA). Similar to Park et al. [26], they limited the search space of the problem for only perpendicular joints architecture.

Lum et al. [28] suggested a new method for kinematic optimization of a spherical mechanism for a minimally invasive surgical robot. The optimization of the mechanism specifically for surgery yields a more compact device than a general spherical manipulator. The optimization balanced between a guaranteed minimum and integrated isotropy over the dexterous workspace as well as total link length in order to yield a very compact, highly dexterous mechanism. Alan et al. [29] introduced a method to optimize the kinematic design of the continuum reconfigurable incision-less surgical parallel (CRISP) robot [30]. A needle-diameter medical robot was proposed based on a parallel structure that is capable of performing minimally invasive procedures. The work maximized the ability of the robot's tip camera to view tissues in constrained spaces. They combine an Adaptive Simulated Annealing (ASA) algorithm, with a motion planner designed specifically for the CRISP robot. ASA facilitates exploration of the robot's design space while the motion planner enables evaluation of candidate designs based on their ability to successfully view target regions on a tissue surface.

Burgner et al. [31] leveraged a grid-based search of the concentric tube continuum robots's configuration space with a nonlinear optimization method to maximize the reachable workspace while satisfying anatomical constraints. Feng et al. [32] optimized laparoscopic port placement for robot-assisted minimally invasive surgery (RMIS) by evaluating the robot's reachable workspace within the patient. Patel and Sobh [33] presented a kinematic optimization framework to find the DH parameters of a robot such that it can reach any task point in a defined task subset. The design objective in this work included the time needed to perform a collision-free motion from an initial position to the target position as well as a dexterity measure to allow for motion corrections in

the neighborhood of the fruits. They used known motion planing algorithms to find the fastest approach.

Others have attempted to find an optimal trajectory for specific tasks. Garg and Kumar [34] proposed a path planning method that searches in the configuration space of a manipulator for an optimal path. The optimal path is based on minimal joint torque and, therefore, leads to low energy consumption. The researchers made use of both GA and SA algorithms and compared performances. The GA algorithm was found to be more efficient. Monteiro and Madrid [35] have used GA to plan the stages of the trajectory of a robot arm. They have proposed the use of GA to plan a trajectory with obstacle avoidance. This is achieved in two stages: initial positioning, which locates the end effector of a robot arm in the initial state, and incremental positioning which moves the end effector to the next state of the trajectory. Pires and Machado [36] have used GA to generate collision free trajectories for robotic manipulators with the objective to minimize the path length and ripple in time evolution of robot positions and velocities. They have used direct kinematics for this purpose and have presented results for several redundant and non-redundant robot manipulators.

2.1.3 Optimization of robotic arms for agricultural applications

For agricultural applications, robotic manipulators are often designed for each task and they aspire to be simple and of low-cost. Scaefe et al. [37] developed an autonomous robot for kiwi harvesting including fruit picking, unloading full bins of fruit, fetching empty bins and protecting the picked fruit from rain. The robot has four custom-designed picking arms programmed to do asynchronously two types of motions in order to pick the fruit. Van Henten et al. [38] used a direct search for optimal design of a cucumber harvesting robot. The design objective included the time needed to perform a collision-free motion from an initial position to the target position as well as a dexterity measure to allow for motion corrections in the neighborhood of the fruits. Classic path planing algorithms (such as RRT, PRM, etc) were used to find the shortest path from the start position to the pick position assuming only the fruit as an obstacle. Bloch et al. [5] develop a methodology for simultaneous optimization of a robot and its working environment and designed an harvesting robot in apple orchards as a test case.

2.2 Evaluation criteria for an optimal arm

Kinematic optimization solutions, such as the ones previously mentioned, use different performance measures to evaluate the kinematics of a robotic arm for general tasks. A common measure is the Manipulability ellipsoid spanned by the singular vectors of the Jacobian and approximates the distance to singular configurations [39]. Similarly, the Global Isotropy Index (GII) measures the ratio between the minimum and maximum singular values of the Jacobian [40]. Another metric approximates the reachable volume of the robot workspace [41]. Paden and Sastry [42] develop relationship between the kinematic performance and the design manipulators with six revolute joints.

The main conclusion from this work reinforces the generally accepted idea that elbow manipulators are good kinematic designs and are optimal with respect to the work-volume. Stock et al. [43] defined space utilization as a new performance index for parallel manipulators. These, however, do not consider tracking accuracy for task-specific paths. Kucuk and Bingul [44] optimized the link lengths and volumes of serial robot manipulators by maximizing the workspace area covered by the robot manipulator and maximized the new local index based on the robot Jacobian matrix. The work showed that spherical robot manipulators have higher global performance indices. The high local performance index does not always produce high global performance index. The new local index, however, cannot be used for comparing the robot manipulators with each other as the global performance index.

2.3 Usage of meta-heuristic algorithms to optimize robot design

Since cost functions that reflect the kinematics of the robot arm are non-linear, gradient-based techniques will most likely fail to find global solutions and, therefore, are not commonly used [17]. On the other hand, nature-inspired meta-heuristic approaches (e.g., PSO, Artificial Bee Colony, Ant Colony Optimization) are more capable in searching for the global optimum in complex problems [45]. For instance, Simulated Annealing was used in the design of a surgical robot to optimize anatomical visibility [29]. GA was used to optimize the design of a manipulator with the GII measure [46]. Similarly, PSO was used to identify the optimal cable routing for a cable-driven manipulator [47]. We observe the performance of these methods for robot design.

2.4 Path planning based on human demonstration

Human demonstrations have been used in robotics for many applications such as imitation learning [48], augmenting reinforcement learning [49] and motion planning [50]. In addition, task-centric optimization was introduced by Kapusta and Kemp [51] to select pose configurations of an existing robot for efficient assistive tasks based on sampled human poses. However and to the best of the author's knowledge, no attempt has been made to optimize the kinematics of a robotic arm for a whole arm tracking of a human demonstrated task. A closely related work by Perez and McCarthy [52] proposed the synthesis of an arm by fitting EE poses on a set of points. Similarly, Shirafuji and Ora [53] proposed kinematic synthesis such that the EE of an under-actuated robotic arm would track some path. However, the derivative of the path must be available at any point for generalized differential Inverse Kinematics (IK). Such derivative cannot be acquired qualitatively from a recorded demonstration. Furthermore, optimization has been attempted to solely decide joint displacements and, did not observe human demonstrations nor considered the minimization of the number of DOF. In addition, having an EE track a path does not ensure that it will be able to prevent collisions of the arm. Hence, we consider the path of the entire arm in order to avoid obstacles in the environment and provide a complete solution.

3 Methodology

3.1 Optimization objective

A task in which an n degrees-of-freedom (DOF) robotic arm must move in an industrial environment with obstacles or in cluttered environment is given. The task could be moving an object from one pose to another (e.g., packaging) or tracking a path (e.g., welding). The assumption is that the environment is static or cluttered with elastic obstacles. An example of elastic obstacles could be tree branches to be moved in order to reach the trunk. The human path is recorded once and, therefore, dynamic obstacles cannot be considered. We aim to find an optimal robotic arm that can perform the task. Without loss of generality, we consider only rotary joints. An optimal robot is the one that incorporates the minimal DOF and provides best accuracy during task execution. We consider position errors for the entire arm from base to end-effector so that arm distance from obstacles can be maximized or the arm can accurately interact with obstacles.

A general optimization of a robotic arm may include kinematic and dynamic design properties. The kinematic properties, which we address in this work, correspond to geometric features and DOF. On the other hand, dynamic properties refer to inertial parameters and depend on various uncertain attributes such as material and fabrication processes of the links; and, internal damping and friction of the actuators. Hence, dynamic properties, unlike kinematic ones, are not easy to optimize. Therefore, we address the optimization problem of the kinematic properties while non-optimal dynamic parameters are commonly compensated using slow motions and control techniques (not in the scope of this work).

3.2 Mathematical Definitions

3.2.1 Tracking Path

As discussed previously, we consider a task in a static or cluttered elastic environment. Hence and in light of the above objective, we seek for a robotic arm with minimal DOF that can accurately track a single path. The single path, in this work, is a path recorded by a human demonstrator. We assume that a path recorded by a human expert is at least near-optimal in the sense that it

is the shortest while maximizing the distance from obstacles or provide the best way to interact with an obstacles in the environment. Hence, the better a robot is able to track the path, the less it is sensitive to positioning errors, risk of collisions and able to push elastic obstacles in the environment.

Given a task, the motion of the human expert arm is recorded. Let m be the number of tracked markers on the arm, a recorded task is given by m paths

$$\mathcal{P} = \{P_1, P_2, \ldots, P_m\} \tag{3.1}$$

where $P_i = \{\mathbf{p}_{i,0}, \ldots, \mathbf{p}_{i,T}\}$ is a path of the i^{th} marker relative to the shoulder. $\mathbf{p}_{i,t} \in \mathbb{R}^3$ is the spatial coordinates of the i^{th} marker at time t (Figure 1.1). We note that P_m is the path of the human hand, which implies about the desired path of the end-effector. We also record a set $\mathcal{V} = \{\mathbf{u}_0, \ldots, \mathbf{u}_T\}$ where elements in $\mathbf{u}_t \in \mathbb{R}^3$ are the Euler angles of the hand or tool at time t. The recorded task can now be used to evaluate the ability of some robot arm to accurately track $\mathcal{P}$.

3.2.2 Robot design variables

To optimize the design of an n-DOF robotic arm, we first define the variables that describe its kinematics. The kinematics of a robot can be defined by a set of coordinates described by the Denavit-Hartenberg (DH) method [54]. In DH, the forward kinematics of an arm is represented by the product of a set of canonical homogeneous transformation matrices $A_i \in SE(3)$ where $i \in \{1, \ldots, n\}$. Transformation matrix A_i defines the position and orientation of the coordinate frame of joint i relative to frame $i-1$ and is conventionally given by

$$A_i = \begin{bmatrix} c(\theta_i) & -s(\theta_i)c(\alpha_i) & s(\theta_i)s(\alpha_i) & a_i c(\theta_i) \\ s(\theta_i) & c(\theta_i)c(\alpha_i) & -cs(\theta_i)s(\alpha_i) & a_i s(\theta_i) \\ 0 & s(\alpha_i) & c(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

where $c(\cdot) = cos(\cdot)$ and $s(\cdot) = sin(\cdot)$. Coordinate frame $\mathcal{O}_b$ is the base frame fixed on the first joint, All the calculation of the robot is taken relative to frame $\mathcal{O}_b$. The four parameters a_i, α_i, θ_i and d_i are standard geometric quantities. Matrix A_i incorporates only one DOF and, thus, depends only on one variable q_i. This variable would be the joint angle $q_i = \theta_i$ for a revolute one. The remaining three are constants and part of the design of the robot. Let $\phi_i \in \mathbb{R}^3$ be a parameterization vector that encodes the parameters of transformation from frame i to frame $i-1$. Hence, ϕ_i is given by

$$\phi_i = (\alpha_i, a_i, d_i). \tag{3.3}$$

Consequently, the product

$$A_{ee}(\mathbf{q}, \Phi) = \prod_{i=1}^{n} A_i(q_i, \phi_i) \tag{3.4}$$

is the robots forward kinematics that express the position and orientation of its end-effector (EE) with regards to the base frame. We define $\mathbf{q} = (q_1, \ldots, q_n) \in \mathcal{C}$ as the vector of joint values

of the arm where $\mathcal{C} \subset \mathbb{R}^n$ is the configuration space of the robot. Vector $\Phi \in \Omega$, where $\Omega \subseteq \mathbb{R}^{3n}$, is the concatenation of all ϕ_i given by $\Phi = (\phi_1, .., \phi_n)$ and is the encoding of the constant parameters. Vector Φ fully-defines the design of a robot and, therefore, Ω is the search space for the optimization problem defined later on.

3.2.3 Allocate robot tracking point

Given a robot defined by Φ, let $\mathbf{s}_\Phi : [0,1] \times \mathcal{C} \rightarrow \mathbb{R}^3$ be a map such that $\mathbf{s}_\Phi(0, \mathbf{q}) = \mathbf{0}$ is the position of the robot base and $\mathbf{s}_\Phi(1, \mathbf{q})$ is the position of its EE. Map $\mathbf{s}_\Phi$ is computed using the forward-kinematics described in Section 3.2.2. Hence, $\mathbf{s}_\Phi(\sigma, \mathbf{q})$ with $\sigma \in [0,1]$ is a parametric function providing the position of any given point σ along a one-dimensional representation of the arm at configuration $\mathbf{q}$ as seen in Figure 3.1a. Function $\mathbf{s}_\Phi$ is non-smooth due to transitions between links at the joints. We also define function $\mathbf{b}_\Phi(\mathbf{q}) : [0,1] \rightarrow \mathbb{R}^3$ which outputs the Euler angles of the EE for robot Φ at configuration $\mathbf{q}$.

The robot links are divided by a user define value ,δ_m , to create a list of points on the tested robot (Figure 3.1b). The robot is divided to m parts, as the number of markers recorded in $\mathcal{P}$. For each robot part, the distance between the marker P_m to the robot point is measured. The marker that has the smallest distance value from the robot is associated to the corresponding point on robot. This process is done for all recorded markers in $\mathcal{P}$.

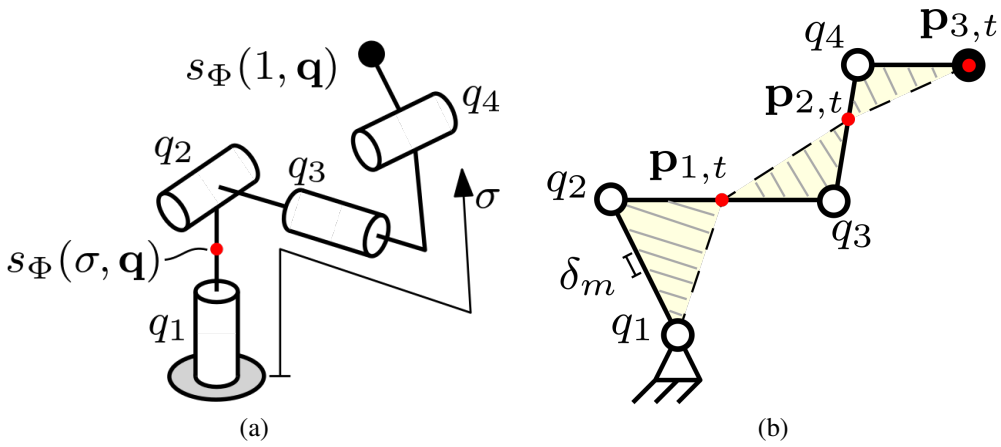


Figure 3.1: (a) Illustration of a robot parameterized by Φ where any point along it can be represented by $\mathbf{s}_\Phi(\sigma, \mathbf{q})$. Points $\sigma = 0$ and $\sigma = 1$ are the base and end-effector points, respectively. (b) Area to minimize (yellow) for a better fit between the robot arm (solid lines) and lines (dashed) formed by the recorded markers (red) where, in practice, we minimize the mean lengths of the grey lines.

3.3 Problem Definition

3.3.1 Robot temporal fitness

A recorded task (3.1) is composed of a sequence of time frames. For each individual frame, we seek to evaluate the fitness of a given robot to the corresponding recorded points. A *temporal fitness* at time t is defined to be the weighted root-mean-square error (wRMSE) between points $\mathbf{p}_{1,t}, \ldots, \mathbf{p}_{m,t}$ and the robot. Therefore, the wRMSE at time t for robot configuration $\mathbf{q}$ is given by

$$g_\Phi(t, \mathbf{q}) = \frac{1}{m}\sqrt{\sum_{i=1}^{m} w_i \|\mathbf{s}_\Phi(\sigma_i, \mathbf{q}) - \mathbf{p}_{i,t}\|^2} + w_0 \|\mathbf{b}_\Phi(\mathbf{q}) - \mathbf{u}_t\|,$$

where parameter σ_i associates $\mathbf{p}_{i,t}$ to the closest point on the robot. Scalars $\{w_0, \ldots, w_m\} \in \mathbb{R}^+$ with $\sum_0^m w_i = 1$ are user-defined weight values that prioritize the position importance of the robot segments relative to the recorded path. In a non-cluttered environment, for instance, the position of the end-effector may have higher significance compared to other links. On the other hand, when working in cluttered environments or when required to push objects with body links, the positions of the body links have higher importance. Hence, we try to find the robot joint values that generate the minimum distance between all the marker to the robot allocated points.

The temporal robot fitness G_t at time t for a robot represented by Φ is given by

$$\begin{aligned} G_{\Phi,t} = \min_{\sigma, \mathbf{q}_t} \quad & g_\Phi(t, \mathbf{q}_t) \\ \text{s.t.} \quad & \mathbf{q}_t \in \mathcal{C}_f \\ & \|\mathbf{q}_t - \mathbf{q}_{t-1}\| \leq \epsilon \\ & \sigma_j < \sigma_k, \ \ \forall j < k \end{aligned} \tag{3.5}$$

where $\sigma = (\sigma_1, \ldots, \sigma_m)^T$, $\mathbf{q}_t \in \mathcal{C}$ is the joint configuration at time t and $\mathcal{C}_f \subset \mathcal{C}$ is the feasible configuration subspace to be discussed below. The second constraint enforces joint path continuity by requiring that two consecutive configurations have a distance smaller than $\epsilon > 0$. Hence, the initial guess for optimization problem (3.5) would be $\mathbf{q}_{t-1}$. Optimization problem (3.5) minimizes the wRMSE of the euclidean distances for all recorded markers from the robot for a single time frame. The solution of problem (3.5) is, in fact, the solution of a multi-point Inverse Kinematics. Next, we use this formulation to calculate the fitness of a robot along the entire path.

3.3.2 Robot path fitness

Given path set $\mathcal{P}$, the *path fitness* $f(\Phi)$ of a query robot design Φ is the mean of the temporal fitnesses given by (3.5) for all time frames $t \in [0, T]$. Formally, a path fitness for robot Φ is defined as

$$f(\Phi) = \frac{1}{T}\sum_{t=0}^{T} G_{\Phi,t}. \tag{3.6}$$

Since we require continuous joint motion and we fix the initial pose of the end-effector based on the task, (3.6) is computed sequentially such that $G_{\Phi,t-1}$ is computed prior to $G_{\Phi,t}$. That is, the joint solution $\mathbf{q}_{t-1}$ of $G_{\Phi,t-1}$ is required in order to solve for $G_{\Phi,t}$ according to (3.5).

Function $f(\Phi)$ minimizes the distance between the robot and recorded markers. However, such minimization is not sufficient since the output may yield a long arm that over-fits the markers as seen on Figure 3.1b. That is, the path fitness can be zero while the arm is very long. Hence, we also minimize the area $E(\Phi)$ (yellow region in the figure) between robot links and represented markers. $E(\Phi)$ is computed as follows. Recall that, at each time frame, the arm is separated into m parts by σ from the solution of (3.5). We divide each part into segments of length δ_m along the robot axis. For each segment, we calculate the distance from the line formed by the two closest markers. $E(\Phi)$ is the mean distance along the robot length and at all time frames. It is important to note that minimizing $E(\Phi)$ indirectly also minimizes robot length $L(\Phi)$.

Finally, the optimal robotic arm Φ^* that can track path $\mathcal{P}$ most accurately is the one with the minimal *robot path fitness* value and is the solution of

$$\begin{aligned} \Phi^* = \quad & arg\min_{\Phi} \quad \lambda_f f(\Phi) + \lambda_E E(\Phi) \\ & \text{s.t.} \quad \Phi \in \Omega_f \end{aligned} \tag{3.7}$$

where $\lambda_f > 0$ and $\lambda_E > 0$ are weighting values. Hence, we wish to minimize the tracking accuracy, link lengths and spatial link movements. Minimization of link lengths will result in a smaller and cheaper arm, along with lower joint loads, i.e., not requiring high-torque actuators. It is important to note that the higher the dimensionality of Φ, i.e., more DOF, it is more likely for the robot to better track the path. However, we seek for a design comprising of minimal DOF. Subset $\Omega_f \subset \Omega$ is the allowed robot design search space defined in the next section.

3.3.2.1 Alternative approach for robot fitness

We have also tested another approach to calculate the *robot path fitness* in order to avoid the dependency of the robot position $\mathbf{q}_{i+1}$ on the previous time joint values $\mathbf{q}_i$. In this approach, we calculate the robot joint value for all time frames at once. The number of optimized parameter is multiplied by the length of the path time frame. For example, if $n = 5$ and the duration of the path is 30 frames, the optimized parameter is 150. In this manner, the joint values for t_{i+1} do not depend on the joint values of t_i. The problem is then given by

$$g^*_\Phi = \frac{1}{T}\sum_{t=0}^{T}\frac{1}{m}\sqrt{\sum_{i=1}^{m} w_i \|\mathbf{s}_\Phi(\sigma_i, \mathbf{q}) - \mathbf{p}_{i,t}\|^2} + w_0\|\mathbf{b}_\Phi(\mathbf{q}) - \mathbf{u}_t\|,$$

$$\begin{aligned} G_{\Phi,T} = \min_{\sigma,\mathbf{q}_t} \quad & g^*_\Phi(q,T) \\ \text{s.t.} \quad & \mathbf{q} \in \mathcal{C}_f \\ & \sigma_j < \sigma_k, \ \ \forall j < k \end{aligned} \quad . \tag{3.8}$$

The robot path fitness is then

$$f^*(\Phi) = G_{\Phi,T}. \tag{3.9}$$

However, Results have shown better performance with regards to computation time and *robot path fitness* while using $f(\Phi)$ of (3.6).

3.3.3 Boundaries and Constraints

This section is define the joint and design search spaces, $\mathcal{C}_f$ and Ω_f, respectively. The allowed joint search space $\mathcal{C}_f$ in (3.5) ensures proper physical movement of a query robot configuration and is defined by the joint limits

$$\mathcal{C}_f = \{\mathbf{q} \in \mathcal{C} |\ \mathbf{q}_{min} < \mathbf{q} < \mathbf{q}_{max}\} \tag{3.10}$$

where $\mathbf{q}_{min}$ and $\mathbf{q}_{max}$ are the minimum and maximum actuators angle limits, respectively. Moreover, to ensure continuous joint motion of the robot $\mathcal{C}_f$ update in each time iteration , t, depends on $\mathbf{q}_{t-1}$ as mention in (3.5).

For any query robot Φ, the world coordinate frame is positioned at the base. The EE coordinate frame is fixed to the last robot link. The static values for the DH parameters in (3.3) are limited according to

$$a_i \in [a_{min}, a_{max}] \text{ and } d_i \in [d_{min}, d_{max}] \tag{3.11}$$

where $0 \leq a_{min} < a_{max}$ and $0 \leq d_{min} < d_{max}$ are pre-defined user variables. Furthermore, we constrain any two consecutive joints to be non-concentric in order to avoid redundancy. Such constraint assists in tightening the search sub-space. The constraint, however, can be omitted with some probability that two robots with n and $n-1$ DOF would have the same fitness due to consecutive joints of the former. A robot solution is considered invalid and omitted if, at time $t = 0$, its total length L does not satisfy

$$L_{min} < L < L_{max} \tag{3.12}$$

where L_{min} and L_{max} are user-defined values. This constraint also aims to filter-out lengthy designs early on. Moreover, if the tested robot configuration cannot reach EE position at $t = 0$ with accuracy of 1 mm, the rest of the path is not calculated. As stated above, the valid search space for $\mathbf{q}_{t+1}$ depends on the previous solution $\mathbf{q}_t$. However, the valid search space for $\mathbf{q}_0$ is defined by the joint angular limits. Hence, the robot can be initialized from different joint values and the rest of the path will be defined accordingly.

Let $\Omega_l \subset \Omega$ be the set of design configurations that satisfy (3.11)-(3.12) and let $\mathcal{Q}_\Phi$ be the workspace of robot Φ such that $\mathbf{s}_\Phi(\sigma, \mathbf{q}) \in \mathcal{Q}_\Phi$ for any $\mathbf{q} \in \mathcal{C}_f$ and $\sigma \in [0, 1]$. Consequently, a valid robot configuration must lie within sub-set $\Omega_f \subset \Omega$ defined as

$$\Omega_f = \{\Phi \in \Omega |\ \Phi \in \Omega_l,\ \mathcal{P} \subset \mathcal{Q}_\Phi\}. \tag{3.13}$$

Subset Ω_f contains all valid robot configurations.

3.4 Optimization algorithm

The definition of the robot fitness and constraints can now be used to search for the optimal design Φ^* of a robot. As discussed in Section 1, kinematic optimization problems are commonly highly non-linear and cannot be efficiently solved using gradient-based techniques. Hence, we employ a meta-heuristic search approach.

Comparative results to be shown in the experiments show that the classic PSO performs best for robot path fitness problem compared to other meta-heuristic optimization algorithms. Consequently, we have proposed a modified algorithm based on the classic PSO. In PSO, N particles $\mathcal{K} = \{\Phi_1, \ldots, \Phi_N\}$ are randomly sampled from a uniform distribution in Ω_l. Each particle Φ_j is evaluated using the `fitness` which returns the robot fitness r_j and the corresponding path $Q_j = \{\mathbf{q}_0, \ldots, \mathbf{q}_T\}$ by solving (3.7). We store and update a personal best position vector bp_j for particle j based on robot fitness. The global best position for all the particles in $\mathcal{K}$ is stored in Φ^* with fitness r^*. Each particle is then updated according to $\Phi_j = \Phi_j + \mathbf{v}_j$ where $\mathbf{v}_j$ is given by

$$\mathbf{v}_j = w\ \mathbf{v}_j + \beta_1 c_1 (bp_j - \Phi_j) + \beta_2 c_2 (\Phi^* - \Phi_j)) \tag{3.14}$$

where constants w, c_1 and c_2 are tunable parameters, and β_1 and β_2 are random numbers in $[0, 1]$. The particles are iteratively updated for I iterations.

Algorithm 1: `RA-PSO`(N, M)

Initialize N agents $\mathcal{K} = \{\Phi_1, \ldots, \Phi_N\} \in \Omega_l$;
for $i = 1 \rightarrow M$ **do**
  for $j = 1 \rightarrow N$ **do**
    $r_j, Q_j \leftarrow$ `fitness`(Φ_j) ; `// sol.` (3.7)
    if $r_j < bp_j$ **then**
      $bp_j \leftarrow r_j$;
  $b \leftarrow arg\min(\{r_1, \ldots, r_N\})$;
  if $r_b < r^*$ **then**
    $\Phi^* \leftarrow \Phi_b,\ r^* \leftarrow r_b,\ Q^* \leftarrow Q_b$;
  $\mathcal{K} \leftarrow$ `update`$(\mathcal{K}, i)$;
return $\Phi^*,\ r^*,\ Q^*$

In order to explore whether we can improve the search of the PSO for our particular optimization problem or to find the solution in less computation effort or commutation time, we propose the RA-PSO algorithm. RA-PSO is a designated variant of PSO, designed to address the properties of search space Φ. Algorithm 1 presents the classic PSO search algorithm which we build upon. RA-PSO maximizes exploitation of particles within Ω_f while avoiding missing-out improved solutions nearby. In order to improve local search, we include two modifications in the update rules

of PSO described in Algorithm 2. First, we distinguish between particles currently in Ω_f and ones that are not. In practice and for computational ease, we approximate the presence of a particle in Ω_f by checking if the temporal fitness at the start poses is smaller than 1 mm. Particles not in Ω_f are updated according to the standard PSO which cause aggressive change in the candidate robot kinematics to explore new regions (line 13). On the other hand, we update a particle Φ_j that is within Ω_f in a significantly lower rate such that its new position is nearby for local refinement and effective exploitation. The velocity $\mathbf{v}_j$ of an agent in Ω_f is updated by adding a random number $[c_{min}, c_{max}]$ multiplied by the boundary range of α_i for angular variables (line 6) and of d_i and a_i for length variables (lines 9 and 11).

The second modification aims to improve local search in Ω_f. Since Ω is large, high-dimensiona and non-linear, we explore different dimensions of Ω by their physical representation. That is, we separately update DH angular (i.e., α_i) and length (i.e., a_i and d_i) variables. In such way, the algorithm is able to better explore subspaces of Ω_f. Let $D \in \mathbb{N}$ be the update frequency of the angular variables. Then, angular variables in Φ_j are updated only when iteration index i can be divided by D. This is implemented using the modulo operator at line 5 of `update` function. This method assists in testing various link lengths with the same angular variables and better improve the exploration of Ω_f.

Algorithm 2: `update`$(\mathcal{K}, i)$

1 **for** $\Phi_j \in \mathcal{K}$ **do**
2 **for** $k = 1 \rightarrow 3n$ **do** // for each dimension in Φ_j
3 **if** $D \neq 1$ *and* $\Phi_j \in \Omega_f$ **then**
4 **if** $mod(i, D) = 0$ **then** // Update either angular or length variables.
5 **if** $mod(k, 3) = 1$ **then** // Update α elements in Φ_j according to (3.3).
6 $v_j^{(k)} \leftarrow v_j^{(k)} + \texttt{random}(c_{min}, c_{max}) \cdot 2\pi$
7 **else**
8 **if** $mod(k, 3) = 2$ **then** // Update a elements in Φ_j according to (3.3).
9 $v_j^{(k)} \leftarrow v_j^{(k)} + \texttt{random}(c_{min}, c_{max}) \cdot (a_{max} - a_{min})$
10 **else** // Update d elements in Φ_j according to (3.3).
11 $v_j^{(k)} \leftarrow v_j^{(k)} + \texttt{random}(c_{min}, c_{max}) \cdot (d_{max} - d_{min})$
12 **else**
13 $v_j^{(k)} \leftarrow w\, v_j^{(k)} + \beta_1 c_1 (bp_j^{(k)} - \Phi_j^{(k)}) + \beta_2 c_2 (\Phi^{*,(k)} - \Phi_j^{(k)})$ // Standard PSO update (3.14)
14 $\Phi_j \leftarrow \Phi_j + \mathbf{v}_j$;
15 **return** $\mathcal{K}$

4 Palm tree leaf model

In this section, we modeled a palm tree as a cluttered environment with elastic obstacles. Figure 4.1 shows workers performing a date thinning task in three different scenarios. For a robot to perform this task and reach a date cluster, it must interact with the tree leaves. Therefore, we aim to understand the behavior of the leaves and model them as elastic obstacles.

Figure 4.1: Plantation workers try to reach a date cluster and perform the date thinning task.

The first thing to be noticed is the connection point between the leaf and the trunk. All the leaf movement is centered around this point. Second, the leaf has five movement primitives: up, down, right, left and the last is to push the leaf towards the trunk, which bends the leaf. After joining the plantation workers and learning the details of the thinning task, it is clear that only the first four primitives described above - up/down and right/left, are necessary to complete the task of reaching the date cluster. With this conclusion in mind along with the leaf connection point, the leaf can be modeled as a hinged rigid beam with torsional springs, as shown in Figures 4.2 and 4.3. In this approach, all biological properties of the leaf can be disregarded.

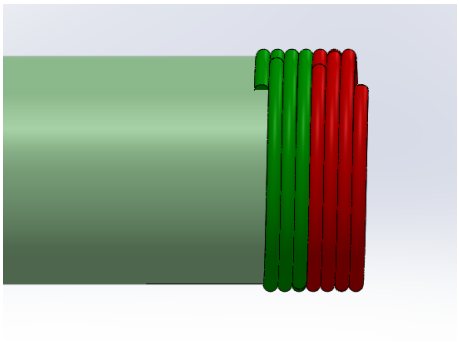


Figure 4.2: Beam with a torsional spring.

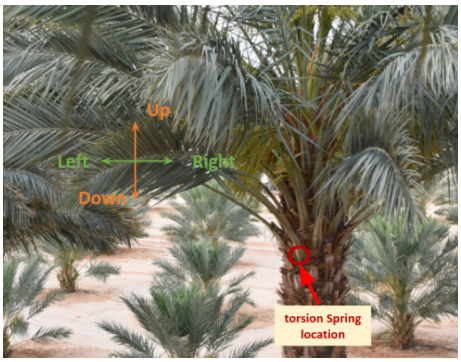


Figure 4.3: Main movement directions of a leaf.

When exerting force onto the beam, the beam rotates around it's base joint. The reaction force from the beam can be calculated according to Hawk's Law. Assuming a known constant spring coefficient K and leaf perturbation Δx, the reaction force is

$$F = K\Delta x. \tag{4.1}$$

This section describes the experiment that was done in order to find the representation of a leaf as a beam with torsional springs at the base. The leaf properties may change with the age of the tree. For example, the older the tree, the longer its leaves become, which affects their mechanical properties. The leaf has four main movement directions. Therefore, we seek for the leaf equation in each direction separately but eventually the leaf can move throughout the whole radius around it's base, and we can perform super-position of vectors to define the force in a specific direction. Another important property of the leaf is that it is anisotropic, due to it's fibrous nature, which means that the leaf is expected to behave differently in different directions. All of these points were taken in consideration during the design of the experiment.

4.1 The experimental apparatus

A crane-like device was designed and fabricated in order to measure accurate force data when applying load on a leaf. The device enables to acquire force measurements while applying various pre-known movements, and enable repeatability in a disorganized environment of the plantation. The device is presented in Figure 4.4. To overcome the uneven land surface, and to verify that the system is leveled, leveling feet were attached to the system. To maintain equal movements in each test, the device was equipped with hooks. By changing the rope configuration, the device has the ability to pull the leaf in all desired directions - up ,down, right and left as presented in Figure 4.5. By using pulleys, the force gauge can indicate the reaction force of the leaf in all different configurations. The combination of the reaction force and the movement of the leaf, when applied to Hook's Law, will provide the spring constant K, for each point along the leaf in each direction.

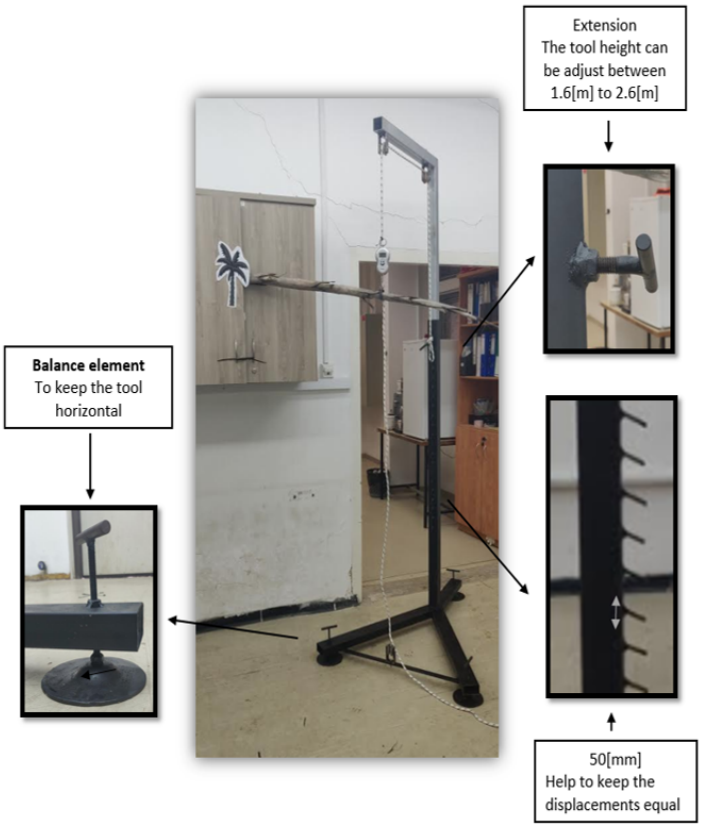


Figure 4.4: The experimental system used to measure the force on a palm tree leaf.

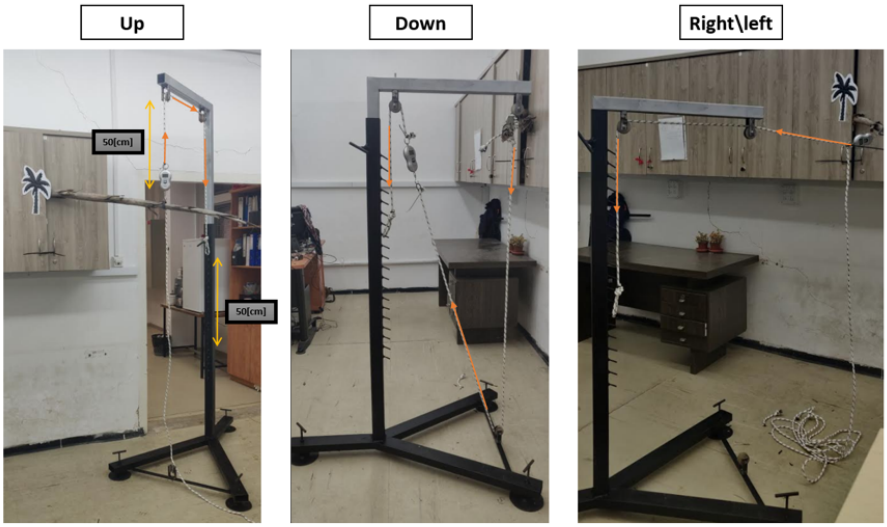


Figure 4.5: Demonstration of the system for measuring reaction forces during the pull of the leaf in all directions.

4.2 Experiment 1- Spring correlation

The first experiment attempts to correlate the changes of leaf behaviour at each point along it to a linear equation (4.1). The experiment was performed on 6 different leaves taken from 6 different trees. The leaves were in the same age-leaf length. Each leaf was approximately $2.5\ m$ of length. The leaves were divided into five segments, by the four different measuring points along them. The points were marked at 20%, 40%, 60%, 80% of the leaf (around 50, 100, 150 and 200 (cm), respectively), as shown in Figure 4.6. To set the measuring point along the leaf, we use

$$distance\_present = \frac{Testing\_point\_distance}{Leaf\_length}. \tag{4.2}$$

The leaf movement was between 5 – 60 cm in each one of the 4 measuring points, and in each direction. Each point on the leaf was measured 12 times in each direction, while increasing the movement by five centimeters each time. The force was measured on the opposing direction to the movement.

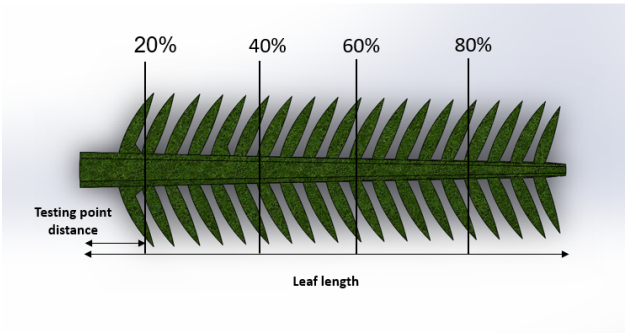


Figure 4.6: Leaf measurement points for experiment 1.

4.2.1 Spring correlation results

The results presented in Figures 4.7-4.10 show a linear correlation between the force and Δx, for each measured point at each direction. Each figure presents the results for all six leaves, for the same distance from the trunk and in the same direction. All the points, in all four directions, show the same linear behavior. This shows that the leaf truly behaves like the model. The average R-square for all measurements is 0.947 (see graph in appendix).

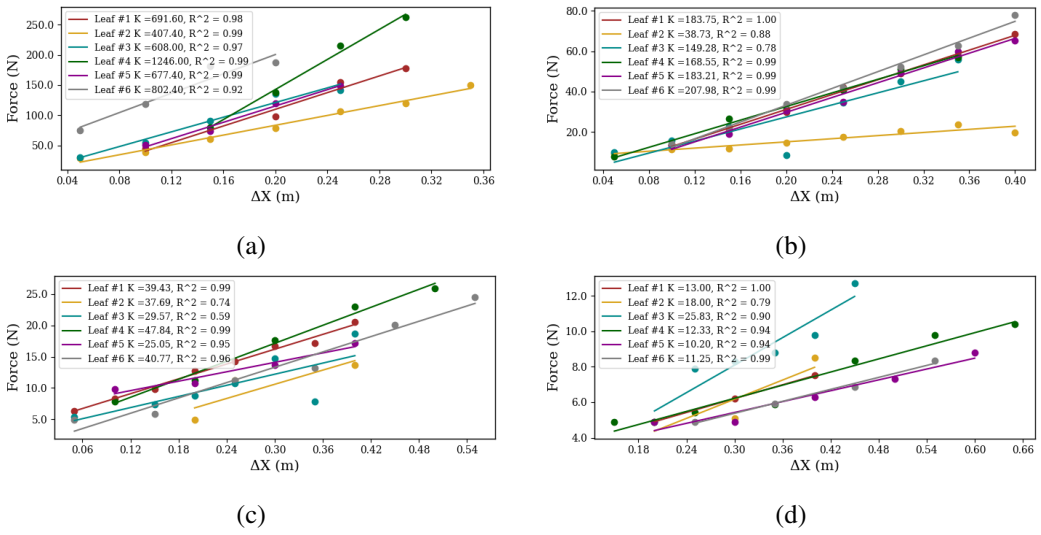


Figure 4.7: Force with respect to Δx of leaf movement in the left direction in 6 different leaves. Distance from the tree trunk is (a) 50, (b) 100 (c) 150 and (d) 200 (cm).

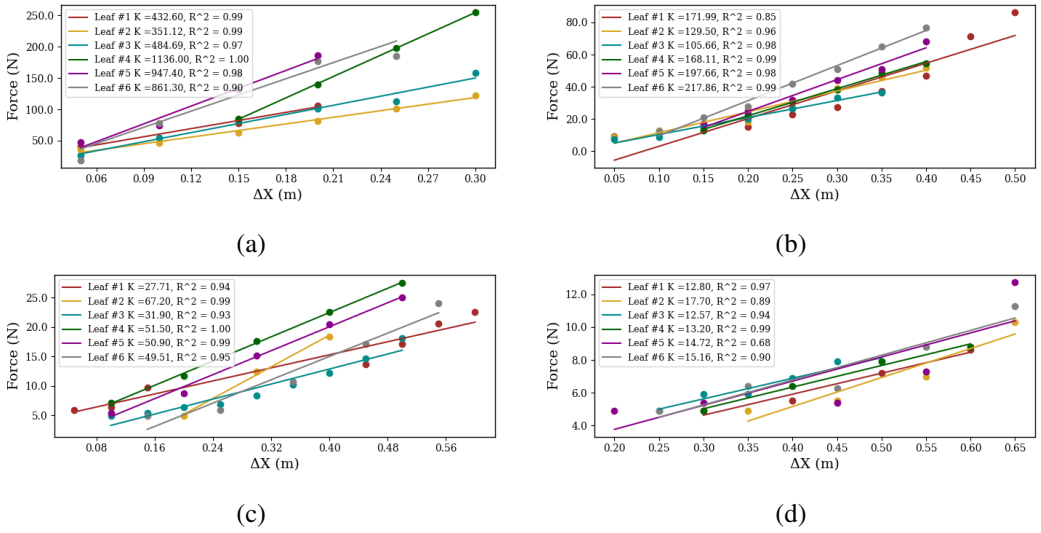


Figure 4.8: Force with respect to Δx of leaf movement in the right direction in 6 different leaves. Distance from the tree trunk is (a) 50, (b) 100 (c) 150 and (d) 200 (cm).

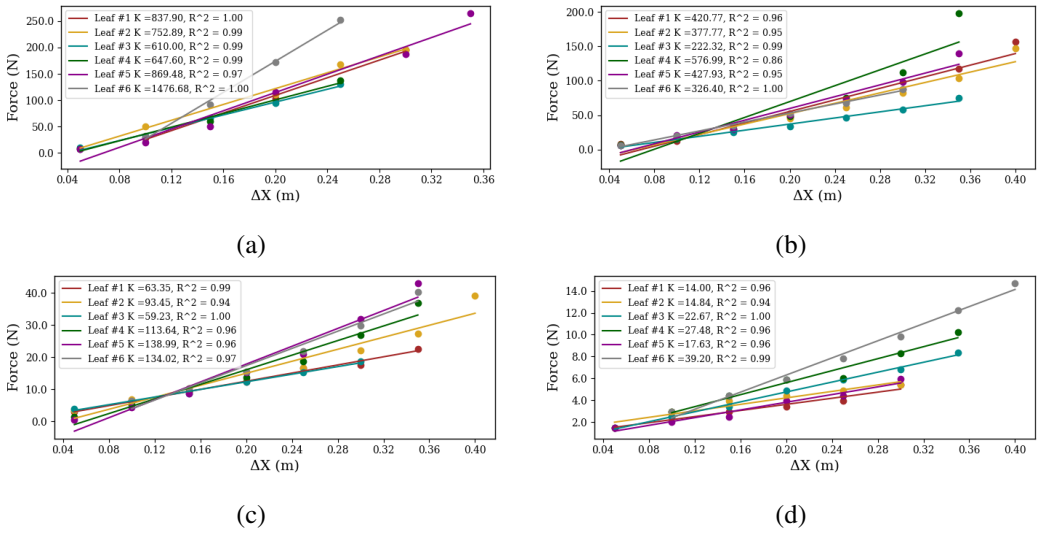


Figure 4.9: Force with respect to Δx of leaf movement in the up direction in 6 different leaves. Distance from the tree trunk is (a) 50, (b) 100 (c) 150 and (d) 200 (cm).

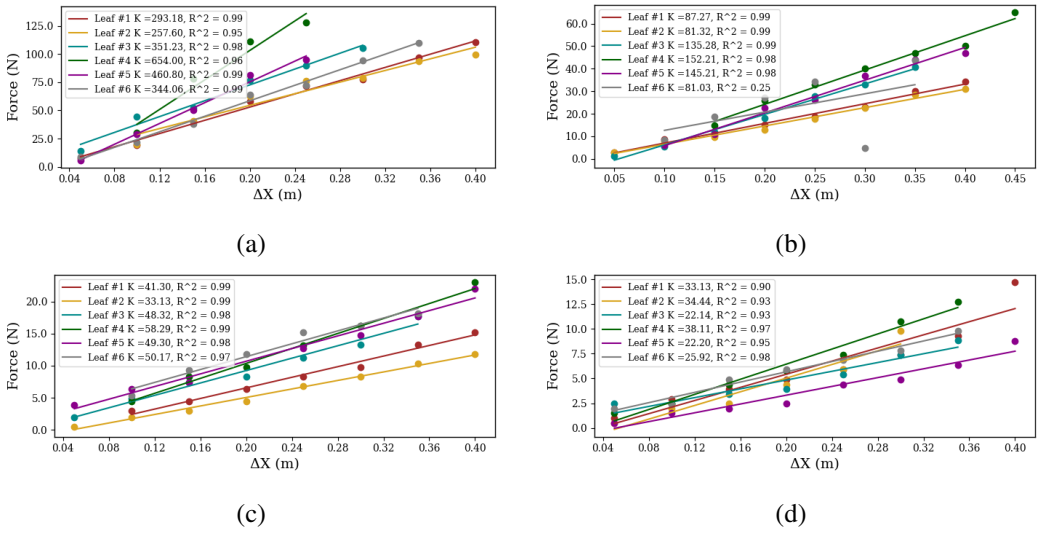


Figure 4.10: Force with respect to Δx of leaf movement in the down direction in 6 different leaves. Distance from the tree trunk is (a) 50, (b) 100 (c) 150 and (d) 200 (cm).

The next step is to understand how the spring constants change along the leaf, and which parameter has the best correlation. Three parameters were checked: normalised distance from the trunk, leaf mass and leaf diameter. The normalised distance was calculated using (4.2). Figure 4.11 shows all values (spring constants) by point normalised distance from the trunk for all directions. After looking at the raw results, it is clear that a linear model is not suitable for describing the spring constants. Exponential equation in the form of

$$y = y_0 e^{-bt} \tag{4.3}$$

is more suitable for this kind of scattering.

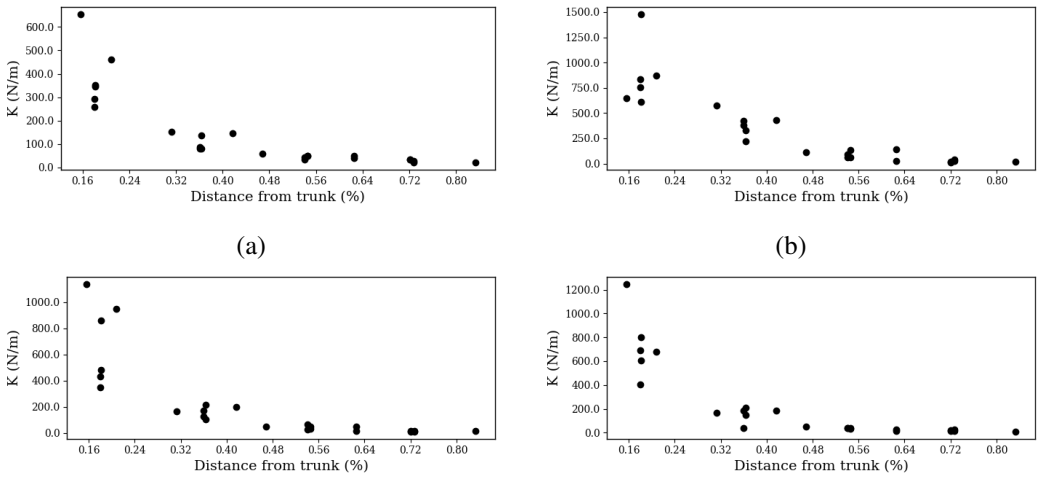


Figure 4.11: Calculated spring constant K with respect to the normalized distance from the trunk for directions (a) up, (b) down, (c) right and (d) left.

A nonlinear exponential regression was performed on the data to find y_0 and b for each parameter (normalised distance from the trunk, leaf mass and leaf diameter) to find the best parameter to characterize the K behaviour along the leaf. The statistical results are shown in Table 4.1. Figure 4.12 presents the results for all tested parameters in all directions. The best parameter to describe the K distribution along the leaf length is the normalised distance from the tree trunk. Moreover, it is the only parameter that can be measured during leaf movement without damaging the tree canopy.

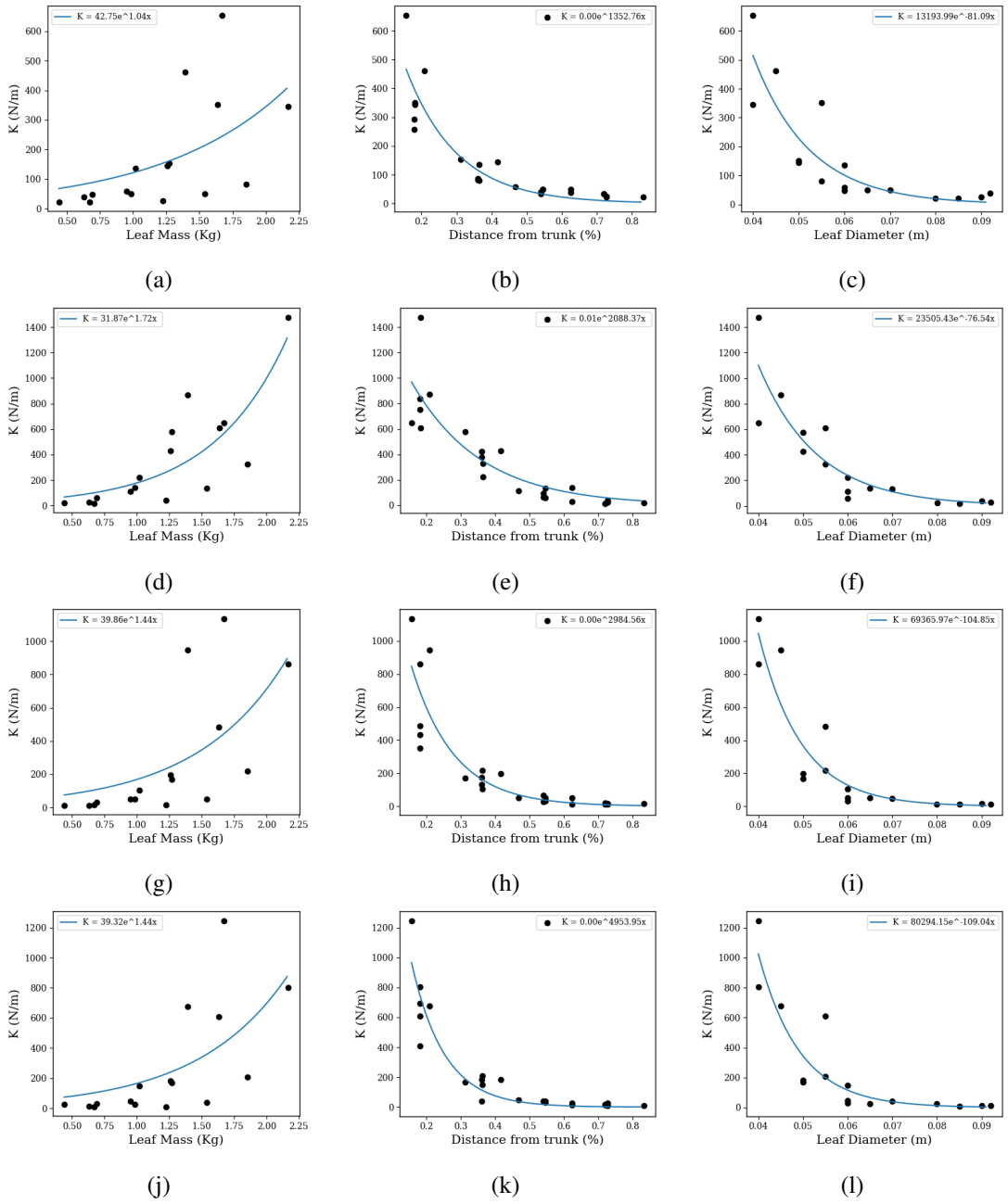


Figure 4.12: Calculated spring constant K by normalized distance from the trunk, leaf mass and leaf diameter for all tested directions down (a,b,c) up (d,e,f) right (g,h,i) and left (j, k ,l). The black dots are the raw results and the blue curves represent the exponential regression results.

Table 4.1: RMSE for nonlinear regression for K behavior along the leaf length in all directions

	RMSE		
	Normelized distance	**Diameter (m)**	**Mass (Kg)**
Up	0.789	0.79	0.67
Down	0.84	0.75	0.30
Left	0.89	0.83	0.41
Right	0.77	0.85	0.41
Mean	0.82	0.80	0.44

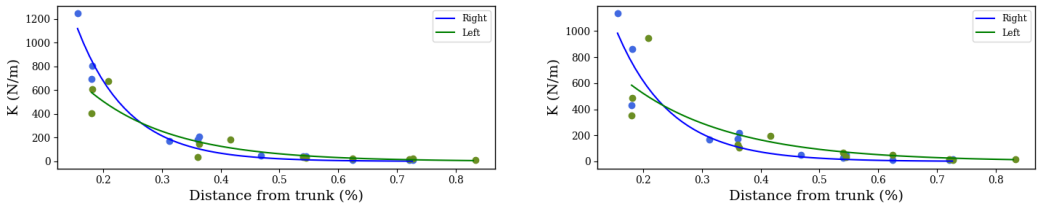
4.3 Right/Left trees

Palm trees can be characterized by the growing direction of their eaves, either right or left. For instance, Figure 4.13 presents a right tree. The direction represents the rotation inclination of the tree. Following the tree rotation, the leaves get the opposite rotation direction and this phenomenon influences their behaviour when an external force is exerted.

We check the influence of the tree rotation on leaf behaviour when force is exerted. We expect to see difference between pulling to the right and left directions - right pulling in a right tree is expected to be easier than pulling to the left direction, and vice versa. The results presented in Figure 4.14 do not support this hypothesis and there is minor force difference between right and left trees.

Figure 4.13: Example of tree rotation to the right.

Figure 4.15: Measuring auxiliary device on the stage.



(a) (b)

Figure 4.14: K (N/m) with regards to the normalised distance from the tree trunk (%) for left (a) and right (b) directional trees. Blue and green lines present the results for right and left trees, respectively.

4.4 Experiment 2 - Adjusting the equations for all leaf lengths

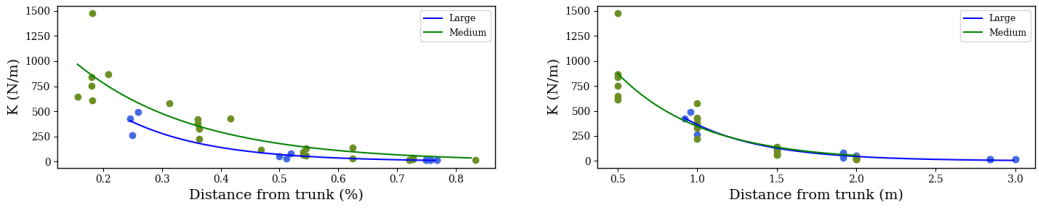
The physical property of the leaf may change with the age of the tree. This section checks the influence of the age and the length of the leaf in mature trees. With such insight, the spring constant calculated with (4.2) in the previous section can be adjusted. A leaf of a mature tree is taken from the treetop at height $18\ m$ above ground. To reach the tree top, the farmer must be assisted with a mast lift in the orchard (Figure 4.15). In mature trees, the trunk has some flexibility, like high building, this property helps the tree withhold from strong winds. The force measurement from this leaf was more noisy due to the movement of the tree and the mast lift in the presence of slight winds.

Table 4.2 presents the results for non-linear regression. Two parameters were checked for correlation: distance and normalised distance (%). The distance is a measure from the tree trunk

Table 4.2: Statistical results for K behavior with exponential correlation for all tested leaf movement directions and lengths

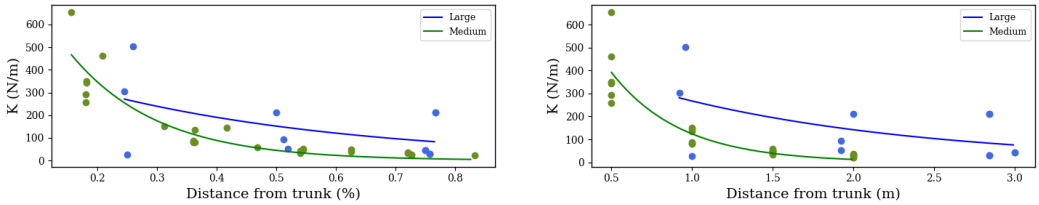
	RMSE	
	Normalized distance	**Distance (m)**
Up	0.78	0.84
Down	0.63	0.56
Left	0.89	0.83
Right	0.78	0.75
Mean	0.77	0.74

in meters, and the normalised distance is computed according to (4.2). The normalized distance shows better statistical results with root mean square error (RMSE) of 0.77. Figures 4.16-4.19 show that the K spring coefficient changes along the leaf with an exponential distribution.



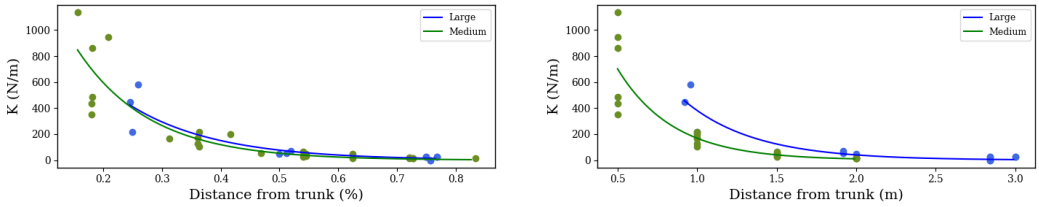
(a) (b)

Figure 4.16: K (N/m) with regards to (a) the normalised distance from the tree trunk (%) calculated according to (4.2) and, (b) the measured distance from the trunk in meters, for the UP direction. Blue curve presents the results for old trees with long leaves and the green curve presents the results for younger trees with medium leaf length.



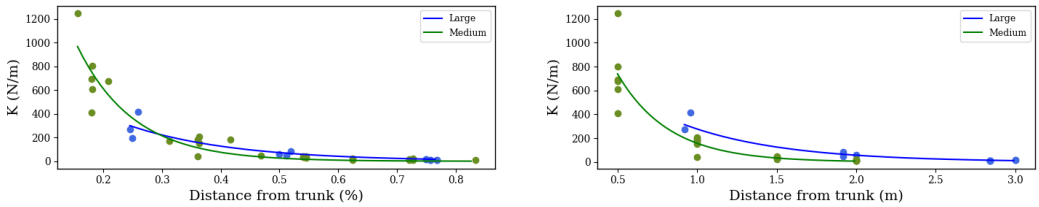
(a) (b)

Figure 4.17: K (N/m) with regards to (a) the normalised distance from the tree trunk (%) calculated according to (4.2) and, (b) the measured distance from the trunk in meters, for the Down direction. Blue curve presents the results for old trees with long leaves and the green curve presents the results for younger trees with medium leaf length.



(a) (b)

Figure 4.18: K **(N/m) with regards to (a) the normalised distance from the tree trunk (%) calculated according to** (4.2) **and, (b) the measured distance from the trunk in meters, for the Right direction. Blue curve presents the results for old trees with long leaves and the green curve presents the results for younger trees with medium leaf length.**



(a) (b)

Figure 4.19: K **(N/m) with regards to (a) the normalised distance from the tree trunk (%) calculated according to** (4.2) **and, (b) the measured distance from the trunk in meters, for the Left direction. Blue curve presents the results for old trees with long leaves and the green curve presents the results for younger trees with medium leaf length.**

4.5 Experiment conclusion - Fit Leaf Equations for all leaf lengths

The results of Experiment 1 and 2 establish a leaf model. The spring constant K exponentially changes along the leaf. The following equations represent the spring constant for each pulling direction.

$$Up : K = 2226.7e^{-5.52x} \tag{4.4}$$

$$Down : K = 977.2e^{-5.13x} \tag{4.5}$$

$$Right : K = 2854.9e^{-7.81x} \tag{4.6}$$

$$Left : K = 5813.3e^{-11.41x} \tag{4.7}$$

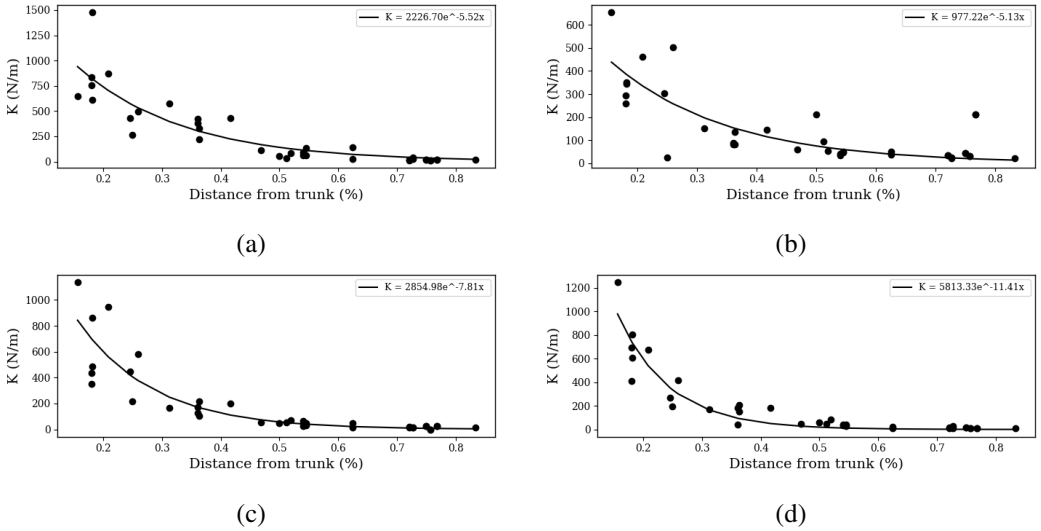


Figure 4.20: Final equations for K in (a) up, (b) down, (c) right and (d) left directions. The black curves represent the correlation and the black dots are from the raw data.

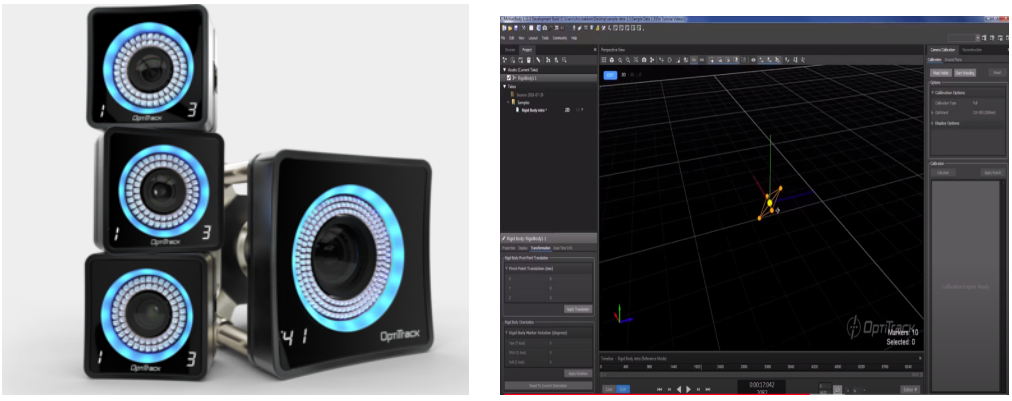
5 Results & Discussion

5.1 Experiments

In this chapter, we will describe the experiments and results to verify and demonstrate the optimization method proposed in Chapter 3.

5.1.1 Setup

Using a motion capture system (OptiTrack system with eight Prime^X-41 cameras) shown in Figure 5.1, we have recorded a human arm performing desired tasks. During recording, four bands are places on the shoulder, upper arm, forearm and hand. Each band has several fiducial markers considered together as a rigid body. Hence, the motion capture system provides real-time data stream of spatial positions of the bands. Data stream may also include orientation of the bands and the hand band in particular. The shoulder is considered as the base position, $\mathcal{O}_b$. The desired robot motion $\mathcal{P}$ is, therefore, the paths of the three distal bands relative to the shoulder band.



(a) (b)

Figure 5.1: (a) Prime^X-41 camera used to record $\mathcal{P}$ and the (b) Motive software used to analyze motion data.

5.1.2 *Test cases*

To validate the proposed method, three test scenarios were considered. For each scenario, a simulated environment was built in the laboratory. In packaging Scenario *I*, an item is picked-up from a conveyor belt, manipulated around an obstacle and placed within a cardboard box (Figure 5.2); In Scenario *II*, welding was performed where a tool must trace a path across an object (Figure 5.3). The figures also show the recorded paths $\mathcal{P}$ of the three distal bands relative to the shoulder band. In Scenario *II*, we constrain the robot end-effector to also trace the orientation of the human hand. Scenario *III* tests a manipulator movement inside a canopy of a palm tree in order to place the manipulator in the position to perform the thinning action of the dates cluster. Chapter 4 presented the results to establish the theory that palm tree leaves act as elastic obstacles during force interaction. In order to create a data tree canopy, a mock-up was built according to the geometry of the Palm tree canopy. The mock-up is presented in Figure 5.4. The leaf holders of this tool were built based on the geometry parameters taken from an adult palm tree as seen in Figure 5.5. No shaking or vibration were observed and, thus, no path smoothing was required. The user-defined parameters and optimization hyper-parameters used for all tests cases are shown in Table 5.1. The hyper-parameters were chosen based on the requirements of the robot in each test case and preliminary experiments.

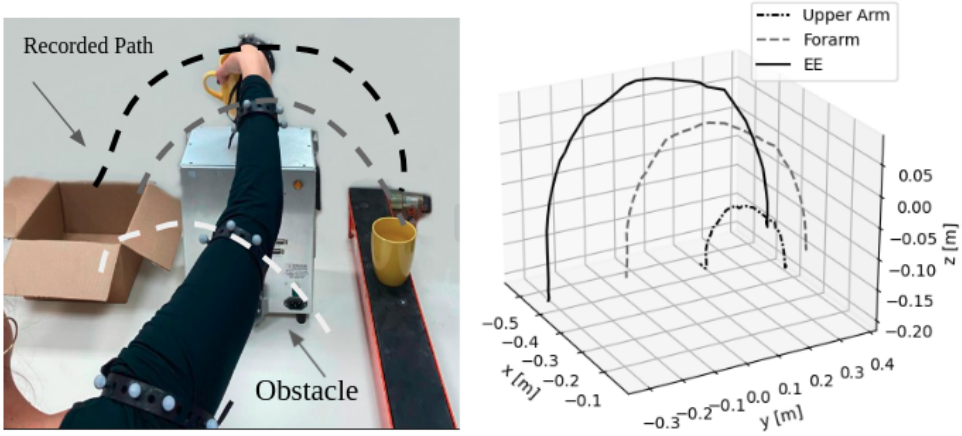


Figure 5.2: Recording of a human arm path in an industrial packaging scenario *I*. A robotic arm is to be optimized to accurately track the recorded path. The end-effector of the robot must follow the path of the hand markers while its arm should follow the human arm markers to avoid obstacles.

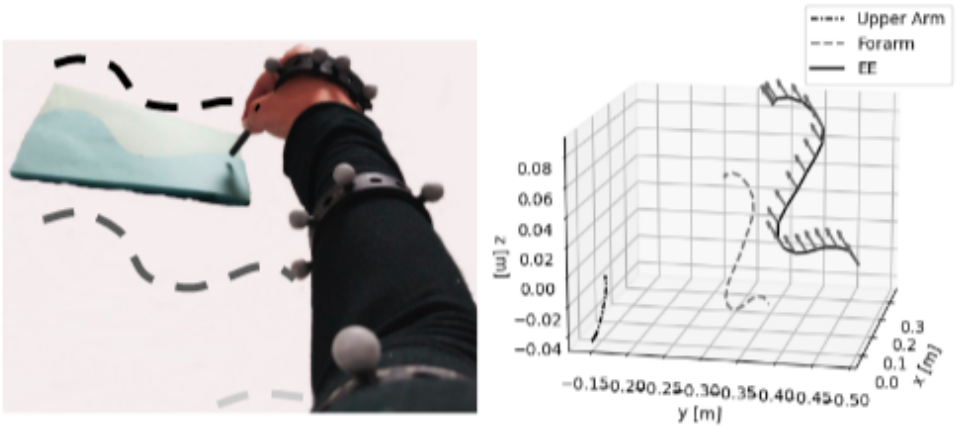


Figure 5.3: Recording of a human arm path in an industrial welding scenario (*II*). A robotic arm is to be optimized to accurately track the recorded path. The end-effector of the robot must follow the path and orientation of the hand markers.

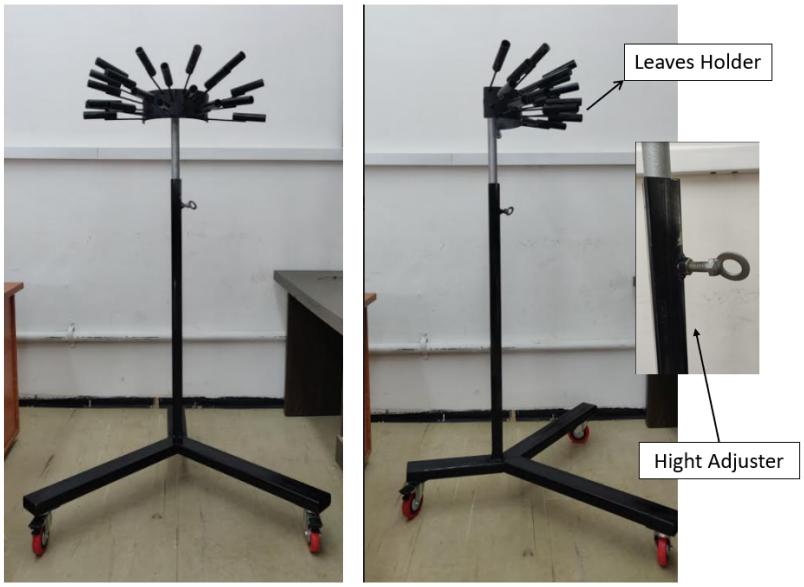


Figure 5.4: Mock-up to simulate a palm tree canopy for recording a date thinning path within the laboratory.

(a) (b)

Figure 5.5: (a) Mock-up to simulate a palm tree canopy. (b) Palm tree leaf growth orientation .

Table 5.1: Optimization parameters

User-defined parameters			
l 0.7 m	L_{min} 0.6 m	L_{max} 1.5 m	ϵ 10 deg
a_{max} 0.5 m	a_{min} 0 m	d_{max} 0.5 m	d_{min} 0 m
q_{max} 180°	q_{min} −180°	δ_m 10 mm	
Optimization hyper-parameters			
w 0.8	c_1 0.4	c_2 0.6	N 400
c_{max} 0.5	c_{min} −0.5		M 200
Scenario *I* - Optimization weights			
w_0 0	w_1 1/6	w_2 1/3	w_3 1/2
λ_f 15	λ_E 5		
Scenario *II* - Optimization weights			
w_0 0.2	w_1 0	w_2 0.1	w_3 0.7
λ_f 15	λ_E 5		
Scenario *III* - Optimization weights			
w_0 0.1	w_1 0	w_2 0.2	w_3 0.7
λ_f 15	λ_E 10		

Table 5.2: Optimization algorithms parameter set-up for solving robot temporal fitness (3.5)

Alg	Particle	Iteretion	Data
PSO	50	50	w =0.8, c1=0.4, c2 = 0.6
ABC	50	50	n_trails =10
ES	50	50	child_ratio= 0.5
WOA	50	50	b = 1
GA	50	50	p_selection =0.75, p_mutation=0.25, p_crossover=0.5
SA	50	50	T = 100, beta = 0.99

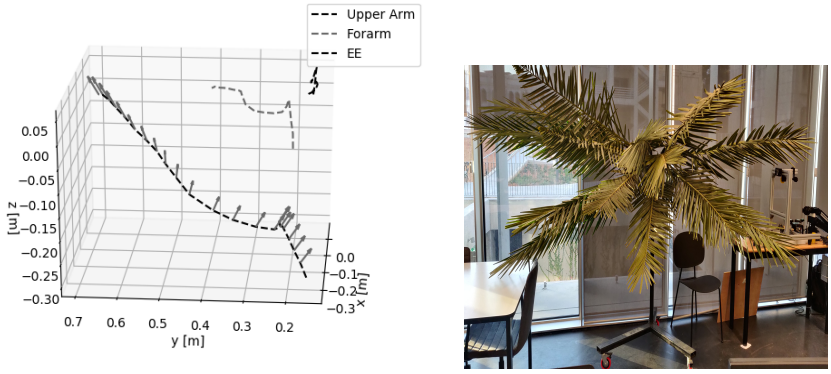


Figure 5.6: (Left) Recording of a human arm path with the palm tree mock-up built in the laboratory (*III*). A robotic arm is to be optimized to accurately track the recorded path. The end-effector of the robot must follow the path and orientation of the hand markers in order to interact with the canopy leaves. (Right) Full tree mock-up built in the laboratory

5.2 Temporal fitness

We first analyze the best algorithm to solve the robot temporal fitness (3.5). We compare between seven different algorithms including PSO, Artificial Bee Colony (ABC), Evolution Strategies (ES), Whale Optimization Algorithm (WOA), Genetic Algorithm (GA), Simulated Annealing (SA) and Sequential Least Squares Programming (SLSQP) [55]. Table 5.2 presents all predefined algorithm parameters. The results of each algorithm are averaged over 30 repetitions and ten different robot configurations in three different time instants along $\mathcal{P}$.

Results of the mean score and time of convergence are summarized in Table 5.3. The means of the standard deviation for all the algorithms are summarized in Table 5.4. Figure 5.7 demonstrates the mean score over time for each tested algorithms. First, SLSQP does not converge well and is, therefore, not suitable for this problem. ABC converges well with low standard deviation and good repetition. However, it has the highest computation time. The remaining algorithms converge relatively similar with small variations about the mean temporal fitness. In particular,

PSO provides somewhat good convergence with the lowest computation time. Hence, the gain in convergence for ABC over PSO is not worth the large effort in computation.

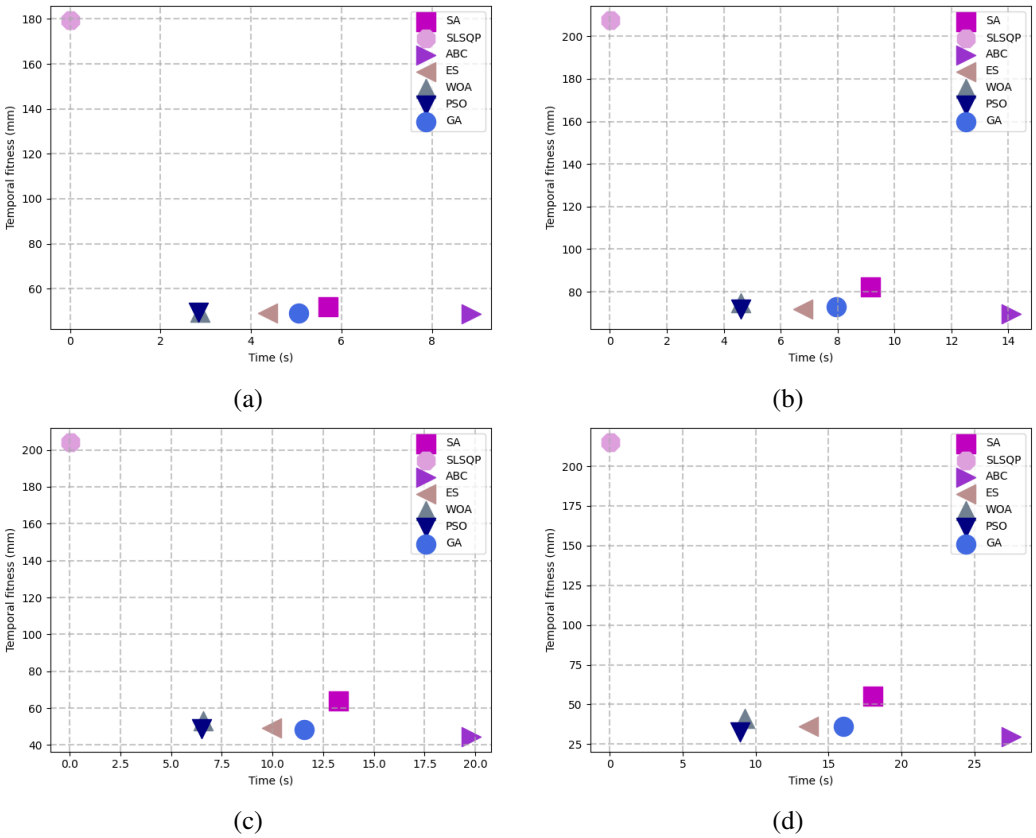


Figure 5.7: Temporal fitness score from (3.5) **with respect to the time of convergence (sec) for all (a)** $n = 3$**, (b)** $n = 4$**, (c)** $n = 5$ **and (d)** $n = 6$**.**

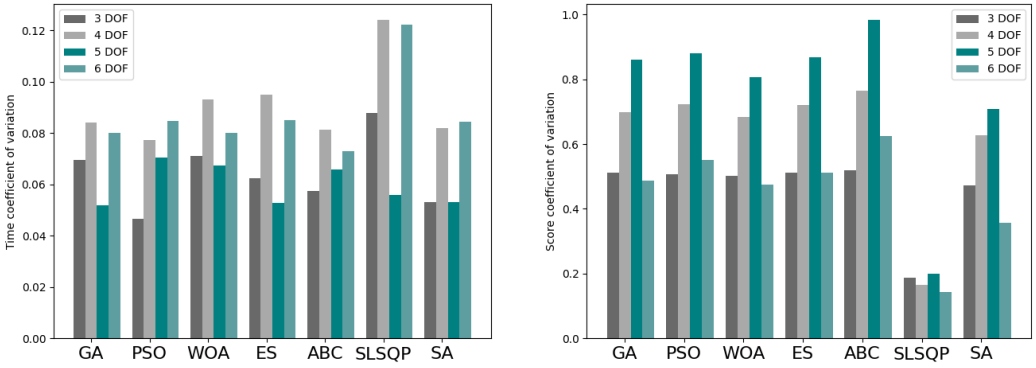
Figure 5.8 shows the coefficient of variation (CV) (the ratio between the standard deviation and the mean) results for the score and time parameters to solve (3.5). For $n = 3, 4$ the CV of the time parameter for PSO shows the lowest results, 0.047 and 0.77, respectively. GA shows the lowest result for $n = 5$. For $n = 6$, ABC shows the lowest result. The difference between ABC and PSO in $n = 5$ and $n = 6$ is 34% and 16%, respectively. However, this improvement is not worth the computation time in our problem. For the temporal fitness score, SLSQP shows the lowest results. However and as shown in Tables 5.3 and 5.4, the method is not suitable for this optimization problem. GA ,PSO, WOA and ABC show the same behavior. Consequently, we choose to use PSO to solve problem (3.7).

Table 5.3: Optimization performance for solving robot temporal fitness (3.5) using various meta-heuristic algorithms

	Mean $G_{\Phi,t}$ (mm)				Median $G_{\Phi,t}$ (mm)				Mean computation time of $G_{\Phi,t}$ (s)			
n	3	4	5	6	3	4	5	6	3	4	5	6
GA	49.04 ±25.10	72.90±50.88	48.50±41.7	36.12±17.62	39.90	51.62	28.33	30.36	5.05	7.96	11.56	15.99
PSO	49.35 ±25.02	71.97 ±52.07	48.76±42.88	32.70 ±17.97	40.11	49.31	26.21	23.60	2.84	4.59	6.51	8.95
WOA	53.13±42.80	74.69 ±50.98	71.80±9.92	40.96 ±19.49	40.78	55.43	32.22	34.18	2.87	4.59	6.58	9.25
ES	48.96 ±25.10	71.93±51.83	49.42±42.87	35.91 ±18.40	39.97	50.56	27.81	28.55	4.37	6.78	9.94	13.62
ABC	48.67 ±25.25	69.51 ±53.22	44.40±43.66	29.52 ±18.46	39.61	49.03	30.67	19.92	8.86	14.09	19.79	27.50
SLSQP	179.25±33.51	207.39 ±34.32	203.92±40.67	214.84±30.97	174.75	206.36	215.00	213.89	7e−3	15e−3	25e−3	39e−3
SA	51.98 ±24.58	82.44±51.74	63.99±45.32	54.97±19.58	41.71	61.18	41.77	45.51	5.70	9.16	13.25	18.05

Table 5.4: Mean of the standard deviation of optimization performance for solving robot temporal fitness (3.5) using various meta-heuristic algorithms

	Mean over Std. of 30 repetitions for $G_{\Phi,t}$			
Alg	3	4	5	6
GA	0.76	3.05	3.65	5.14
PSO	2.07	4.48	6.57	6.79
WOA	1.95	4.95	7.34	8.57
ES	0.79	2.82	5.23	5.63
ABC	0.008	0.86	2.50	3.22
SLSQP	60.35	55.04	48.87	71.00
SA	4.34	8.89	10.03	12.10



(a) (b)

Figure 5.8: (a) Coefficient of variation for the time parameter. (b) Coefficient of variation for the score parameter.

5.2.1 Robot path fitness - $f^*(\phi)$

This experiment goal is to calculate the robot joint value for all time frames at once using equation (3.8), $g^*(\phi)$. The number of optimization parameters is multiplied by the length of the path time frame. For example, if $n = 5$ and the duration of the path is 30 frames, the dimension of the optimization vector is 150. In this manner, the joint values for t_{i+1} do not depend on the joint values of t_i.

We compare the results to (3.7) ($g(\phi)$). As mentioned in the previous section, in order to calculate $f(\phi)$ we iterate over time and set new joint limits for each iteration based on the previous robot state. A single robot was tested 10 times with the same path $n = \{4, 5, 6\}$. To solve $f^*(\phi)$, we use $I = 500$ and $N = 1000$. For $f(\phi)$, we use $M = 150$ and $N = 100$. For both, $c_1 = 0.4$, $c_2 = 0.6$ and $w = 0.8$ are used. The results in Figure 5.9 show that $f(\Phi)$ is better by a scale of 1.48, 2.08 and 2.14 for $n = 4$, $n = 5$ and $n = 6$, respectively. Figure 5.10 presents the computation time results. The computation time with $f^*(\Phi)$ is significantly larger by 31.35%. The computation time has a large impact and we choose to continue our tests with $f(\phi)$.

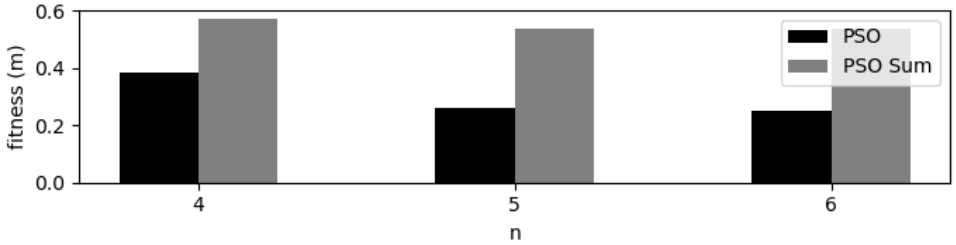


Figure 5.9: Mean results of 10 repetitions for $f(\phi)$ and $f^*(\phi)$ with a full path and with regards to $n = 4...6$.

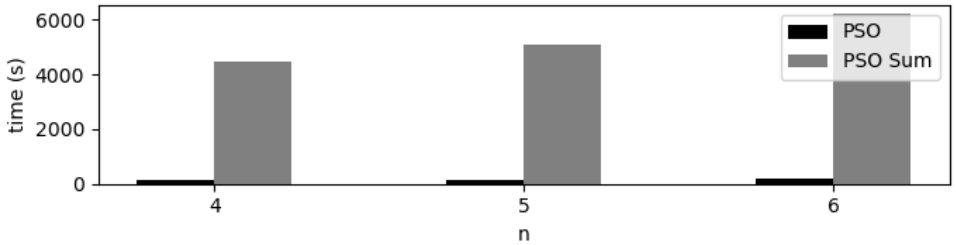


Figure 5.10: Mean time of 10 repetitions for $f(\phi)$ and $f^*(\phi)$ with a full path and with regards to $n = 4...6$.

5.3 Robot fitness tests with different algorithms

This section analyzes the best algorithm to solve robot fitness (3.7). We analyze the algorithms for scenario *I*. Furthermore, we compare between four different algorithms including PSO, Artificial Bee Colony (ABC), Genetic Algorithm (GA) and Simulated Annealing (SA). Set-up parameters for each algorithm are summarized in Table 5.5. The results for each algorithm are averaged over 30 repetitions and presented in Table 5.6.

Table 5.5: Optimization algorithms parameter set-up for solving robot fitness Φ^*

	particle	Iteration	data
ABC	400	200	n_trials: 10
GA	400	200	p_selection:0.75, p_mutation:0.25, p_crossover: 0.5
PSO	400	200	w: 0.8, c1: 0.6, c2: 0.4
SA	400	200	T : 100, beta: 0.99

PSO presents the best mean results for Φ^* with $n = \{3, 4, 5, 6\}$. On the other hand, SA computation time is lower from the PSO for $n = 3$ but the score of the PSO is better in 40% for the SA. ABC has a standard deviation lower by 70% than the PSO. However, the mean iteration time is 2.35 times than the PSO. PSO also provides the best results for $f(\Phi)$ and $E(\Phi)$ with $n = \{3, 4, 5, 6\}$ compared to all tested algorithms. Moreover, PSO shows the lower sensitivity to increasing $n = \{4, 5, 6\}$ in all tested parameters. In other words, PSO can manage a large number of optimization parameters (larger search space) and find the best results. The results for $n = 3$ behave differently for all parameters and algorithms. The reason is assumed to be the lack of enough DOF to track the complex recorded path.

ABC and GA require the largest computation time compared to PSO and SA. SA scores slightly higher than the PSO in all the parameters for all tested n with slightly lower standard deviation than the PSO. Therefore, the added value of the ABC algorithm is not worthy for our optimization problem and we choose the PSO to solve for Φ^*.

Table 5.6: Mean results for optimization algorithms solving robot fitness Φ^*

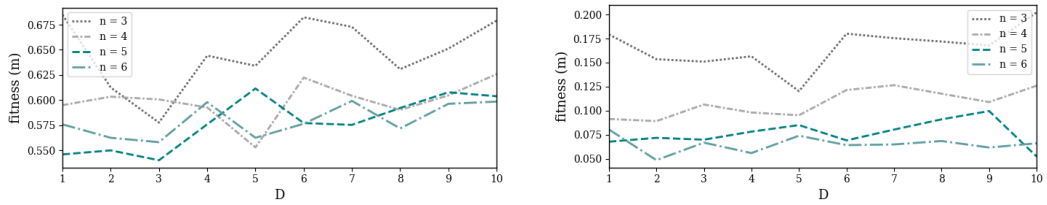
		Mean±Stdev			
		3	4	5	6
Φ^*	ABC	0.86 ± 0.10	0.62±0.05	0.57±0.03	0.51±0.04
	GA	1.03±0.20	0.62±0.08	0.6±0.06	0.58±0.04
	PSO	0.59±0.17	0.5±0.06	0.48±0.08	0.48±0.06
	SA	1±0.09	0.67±0.11	0.6±0.05	0.57±0.04
Time (h)	ABC	4.24±0.83	3.21±0.81	3.37±0.37	3.52±0.42
	GA	5.26±2.50	4.38±1.76	3.47±3.61	2.91±1.06
	PSO	1.80±1.27	0.58±0.17	0.63±0.34	0.64±0.15
	SA	0.72±0.08	1.16±0.14	1.65±0.25	1.84±0.25
$f(\Phi)$**(mm)**	ABC	35.16±5.34	26.62±2.76	23.90±1.03	20.69±2.05
	GA	46.48±10.32	24.92±3.29	25.66±2.99	23.82±2.94
	PSO	23.92±9.30	20.39±3.01	19.36±4.37	19.12±3.83
	SA	44.62±6.33	27.67±5.94	24.17±2.60	24.52±1.54
$E(\Phi)$ **(mm)**	ABC	65.86±11.47	43.55±2.64	41.79±3.88	40.56±4.10
	GA	66.63±13.74	49.70±8.32	43.28±5.95	45.08±3.45
	PSO	46.93±10.60	38.92±4.38	38.40±5.11	38.44±3.61
	SA	66.63±8.70	50.01±8.64	46.43±5.59	41.29±5.25
$L(\Phi)$ **(m)**	ABC	0.81±0.10	0.73±0.02	0.72±0.02	0.73±0.02
	GA	0.76±0.05	0.77±0.06	0.73±0.02	0.75±0.03
	PSO	0.74+0.05	0.72±0.02	0.72±0.02	0.72±0.02
	SA	0.78±0.05	0.78±0.05	0.75±0.03	0.73±0.02

5.4 RA-PSO

This section tests the proposed RA-PSO algorithm. First, we test the value of D (the update frequency of the angular variables) for scenario *I* and compare all the tested parameters. Second, we compare the RA-PSO performance for scenarios *II* and *III*. We test all scenario convergence iterations and Ω_f. All the results in this section are the mean results for 30 repetitions of each test.

5.4.1 Find D limits

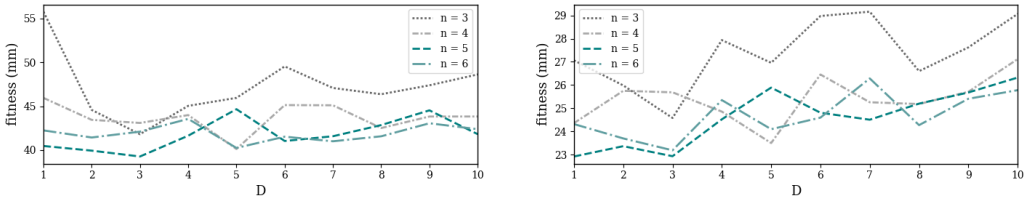
Figure 5.11 presents Φ^* for scenario *I* with regards to D. For $n = 3$ we see that Φ^* decreases until $D = 3$ and increases for $D = 4, 5, 6$. The optimization for $n = 4$ behaves the same for $D = 2, 3, 4$ and decreases for $D = 5$. When $D > 5$, the robot fitness Φ^* has a trend upwards. For $n = 5, 6$ and $D = 2, 3$, increasing D does not impact the final result of Φ^*. The standard deviation with regards to D is relatively constant as shown in 5.11b. Moreover, the standard deviation decreases when n increases.



(a) (b)

Figure 5.11: (a) Mean and (b) standard- deviation results for Φ^* with $n = 3, 4, 5, 6$ and $D = 1, 2, \ldots, 11$ over 30 repetitions.

Figure 5.12 presents the mean score of $f(\Phi)$ and $E(\Phi)$ for 30 repetition while preforming scenario *I*. Φ^* behaves the same as $f(\Phi)$. We expect this kind of behavior due to the fact that λ_f is 75% of the calculation of Φ^*. $E(\Phi)$ behave the same for $D = 2, 3, 4$ for $n = 4, 5$ with low variance for D. The largest improvement in $E(\Phi)$ is for $n = 3$ for $D = 2, 3, 4, 5$. For $n = 6$ there is a step up in the results from $D > 4$. Taking all in consideration, we choose to continue our testing all scenarios for $D = 2, 3, 4, 5$.



(a) (b)

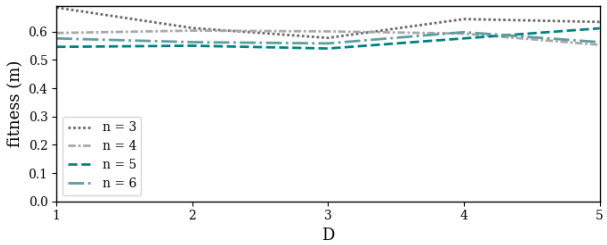
Figure 5.12: Mean results for (a) $E(\Phi)$ and (b) $f(\Phi)$ for $n = 3, 4, 5, 6$ and $D = 1, 2, \ldots, 11$ over 30 repetitions.

5.4.2 Compare all scenarios optimization results

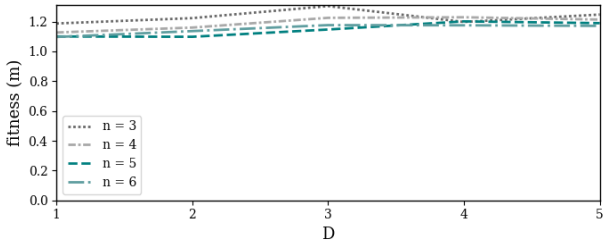
Figure 5.13 presents the robot fitness result with regards to D for all scenarios and n. Table 5.7 presents the ratio for the robot fitness for $D = 2, 3, 4, 5$ for all n. The ratio is calculated in the following form

$$\Delta = \frac{Result_{D=1} - Result_{D=i}}{Result_{D=1}}. \tag{5.1}$$

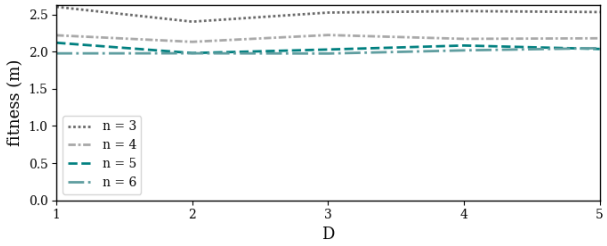
Negative results indicate improvement while positive suggests that the PSO performs better than the RA-PSO. The results imply that the complexity if the recorded path $\mathcal{P}$ has impact on the RA-PSO results. We sort the difficulty of the tested robot path $\mathcal{P}$ from the easiest (scenario *I*) to the hardest (scenario *III*). As shown in Figure 5.13, a robot with larger n scores less than a robot with lower n. Moreover, we can see that, for scenario *III*, the relative error from the classic PSO in the performance of Φ^* is between 7% improvement to 4% reduction. For scenario *I* the results show larger variation between 15% improvement to 12% reduction.



(a)



(b)



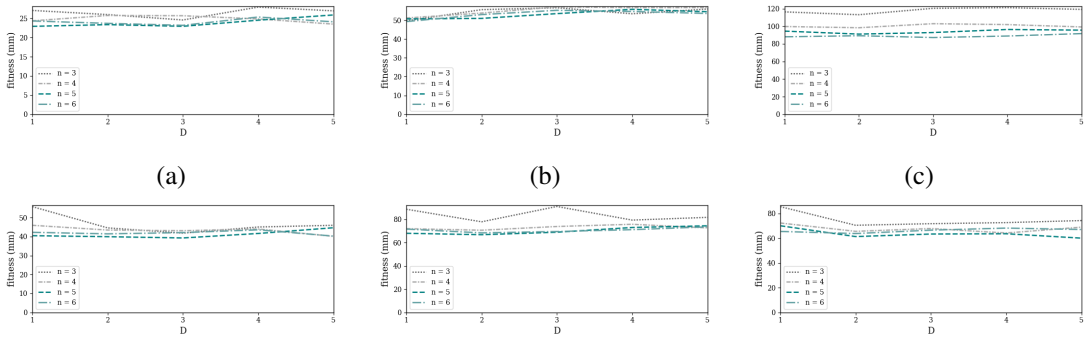
(c)

Figure 5.13: Robot Fitness (3.7) **with** $D = 2, 3, 4, 5$ **and** $n = 3, 4, 5, 6$ **for scenarios (a)** ***I*****, (b)** ***II*** **and (c)** ***III*****.**

For scenario *III* and $n = 4, 5, 6$, the RA-PSO improves the results for $E(\Phi)$. For scenario *II*, $E(\Phi)$ presents low performance than the classic PSO. The reason for this behavior is the complexity of $\mathcal{P}$ and the set-up of the w_n weights. In this path, the robot is ordered to also trace the orientation of $\mathcal{P}$. This has impact on the ability of the robot to track $f(\Phi)$ and we set a lower weight to $E(\Phi)$.

Table 5.7: Percentage (%) of improvement for $D = 2, 3, 4, 5$ for all n and all scenarios. Negative score implies improvement from the classic PSO.

		Φ^*				$f(\Phi)$				$E(\Phi)$			
		D				D				D			
n	scenario	2	3	4	5	2	3	4	5	2	3	4	5
3	*I*	-10.54	-15.64	-5.92	-7.4	-3.99	-9.19	3.22	-0.39	-20.07	-25.04	-19.24	-17.61
	II	3.03	9.84	0.86	5.04	12.08	14.11	7.65	12.74	-12.16	2.66	-10.51	-7.87
	III	-7.64	-2.95	-2.17	-2.69	-2.76	3.49	4.16	2.44	-17.59	-16.08	-15.08	-13.14
	Mean Improvement(%)	-5.05	-2.92	-2.41	-1.68	1.78	2.80	5.01	4.93	-16.61	-12.82	-14.94	-12.87
4	*I*	1.41	0.95	-0.38	-7.05	5.75	5.46	2.04	-3.51	-5.44	-6.21	-4.26	-12.68
	II	2.92	8.73	9.06	7.69	5.15	11.59	10.83	10.87	-18.2	2.63	5.27	0.91
	III	-3.9	0.15	-2.17	-1.83	-1.39	3.21	2.19	-0.47	-9.32	-6.18	-11.18	-4.62
	Mean Improvement(%)	0.14	3.28	2.17	-0.40	3.17	6.75	5.02	2.30	-10.99	-3.25	-3.39	-5.46
5	*I*	0.73	-1.06	5.47	12	1.93	0.05	6.99	12.96	-1.31	-2.96	2.89	10.36
	II	-0.18	4.24	9.21	8.13	0.54	5.54	10.04	7.54	-1.8	1.34	7.34	9.45
	III	-6.52	-4.3	-1.74	-3.95	-3.57	-1.7	1.96	1.18	-12.48	-9.54	-9.24	-14.3
	Mean Improvement(%)	-1.99	-0.37	4.31	5.39	-0.37	1.30	6.33	7.23	-5.20	-3.72	0.33	1.84
6	*I*	-2.31	-3.07	3.83	-2.31	-2.54	-4.65	4.25	-0.9	-1.91	-0.34	3.1	-4.73
	II	3.59	7.25	7.13	6.78	7.54	12.18	10.93	8.76	-4.55	-2.93	-0.69	2.67
	III	0.05	-0.09	2.03	3.55	1.36	-0.94	0.98	4.12	-2.58	1.59	4.14	2.41
	Mean Improvement(%)	0.44	1.36	4.33	2.67	2.12	2.20	5.39	3.99	-3.01	-0.56	2.18	0.12



(a) (b) (c)

(d) (e) (f)

Figure 5.14: Results for $f(\Phi)$ and $E(\Phi)$ with $D = 2, 3, 4, 5$, $n = 3, 4, 5, 6$. Results for $f(\Phi)$ in scenarios (a) *I*, (b) *II* and (c) *III*. Results for $E(\Phi)$ in scenarios (d) *I*, (e) *II* and (f) *III*.

To summarize the results of the RA-PSO, we can see that the differences of all $D = 2, 3, 4, 5$ and $n = 3, 4, 5, 6$ between the classic PSO and the RA-PSO is low (lower than 10%). Finally, we can say that the results of the RA-PSO are the same as the classic PSO and we need to find other way to check if the RA-PSO is worthy.

5.4.3 Convergence iteration and valid particles

This section test the number of iterations to convergence and the number of valid particles in Ω_f for $D = 2, 3, 4, 5$. Due to the fact that we run all the tests on different capacity of the computer, we cannot compare the iteration time and we seek for new parameter to evaluate the RA-PSO performance.

First, we check the mean number of iterations for convergence for 30 repetitions. The results shown in Figures 5.15 and 5.16 are the mean number of iterations for convergence M_i for all and each scenario, respectively, with respect to parameter D. Note that $D = 1$ is the regular PSO algorithm. Hence, for $n = 3$ and $D = 2, 3$, converge is 13% faster than the regular PSO. For $n = 4$ and $D = 3$, converge is faster by 25%. The difference between $D = 2, 4, 5$ is lower than 1%. For $n = 5$ and $D = 2, 3, 4, 5$, converge is faster but the largest impact is for $D = 5$ (29%). For $n = 6$ and $D = 4$, the convergence is faster by 55%. All the results for the improvement in convergence for all n and all tested D are presented in Table 5.8.

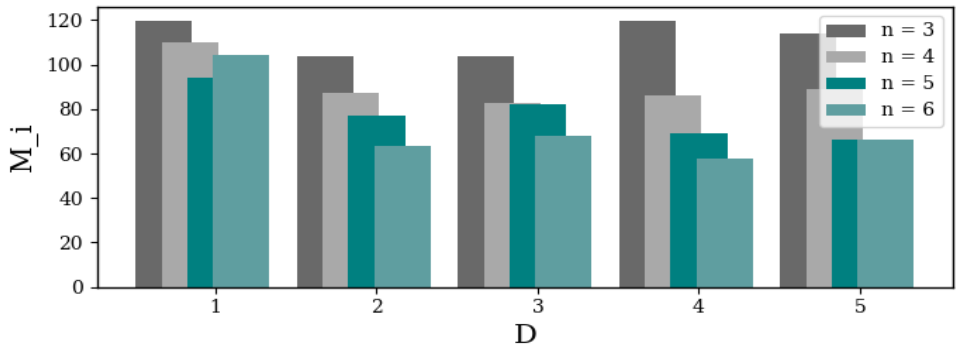
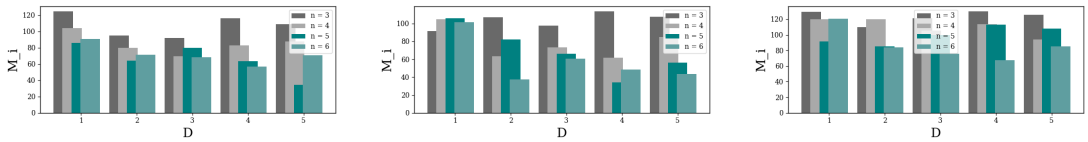


Figure 5.15: Mean number of iterations to convergence for all scenarios . $1 \leq D < 6$ **,** $n = 3, 4, 5, 6$



(a) (b) (c)

Figure 5.16: Mean convergence iteration . $1 \leq D < 6$ **,** $n = 3, 4, 5, 6$ **for scenario *I* scenario *II* and scenario *III*,(a),(b),(c) respectively.**

Figure 5.17 presents the mean valid agents found in Ω_f in each iteration for all scenarios and for $n = 3, 4, 5, 6$. Figure 5.18 presents the results for each scenario separately. The percentage of improvement from the classic PSO is presented in Table 5.9. All scenarios exhibit the same behavior for all n. Moreover, it seems that the valid agents for all scenarios for all D are decreasing while increasing n. It is assumed that the size of Ω_f has large impact on the number of particles in Ω_f found in each iteration. For all scenarios, the classic PSO with $n = 3$ shows a unique number of valid agents found during each iteration. Using RA-PSO yields a more accurate search in Ω_f while keeping the same objective function fitness as discussed in Section 5.4.2. The classic PSO have the largest number of valid particles (Ω_f) but it converges in a higher number of iterations with almost the same results as the RA-PSO.

Table 5.8: Improvement in percentage (%) from classic PSO and converge iteration result for $D = 2, 3, 4, 5$ for all n all scenarios, Negative results implied on improvement from the classic PSO.

		Iteration number improvement (%)				Iteration number score				
		D				D				
n	scenario	2	3	4	5	1	2	3	4	5
3	*I*	-23.93	-26.15	-6.58	-12.29	124.67	94.83	92.07	116.47	109.34
	II	16.59	6.47	23.86	17.46	91.69	106.90	97.62	113.57	107.70
	III	-15.07	-6.33	0.36	-2.92	129.40	109.90	121.21	129.86	125.62
	Mean Improvement(%)	-7.47	-8.67	5.88	0.75	115.25	103.88	103.63	119.97	114.22
4	*I*	-23.23	-33.51	-20.58	-15.59	104.30	80.07	69.34	82.83	88.03
	II	-39.73	-30.17	-41.32	-19.31	105.13	63.37	73.41	61.69	84.83
	III	0.00	-12.46	-5.41	-21.79	120.14	120.14	105.17	113.63	93.96
	Mean Improvement(%)	-20.99	-25.38	-22.44	-18.90	109.86	87.86	82.64	86.05	88.94
5	*I*	-25.69	-7.35	-26.11	-60.26	86.17	64.03	79.83	63.67	34.24
	II	-22.67	-37.41	-67.64	-47.22	106.04	82.00	66.37	34.31	55.97
	III	-6.85	8.43	23.36	17.49	91.75	85.46	99.48	113.19	107.80
	Mean Improvement(%)	-18.40	-12.11	-23.46	-30.00	94.65	77.16	81.89	70.39	66.00
6	*I*	-21.32	-24.81	-37.06	-21.87	90.79	71.43	68.26	57.14	70.93
	II	-63.33	-40.29	-52.02	-56.91	101.34	37.17	60.52	48.63	43.67
	III	-30.59	-37.28	-44.21	-29.51	120.48	83.63	75.57	67.21	84.92
	Mean Improvement(%)	-38.41	-34.13	-44.43	-36.10	104.20	64.08	68.12	57.66	66.51

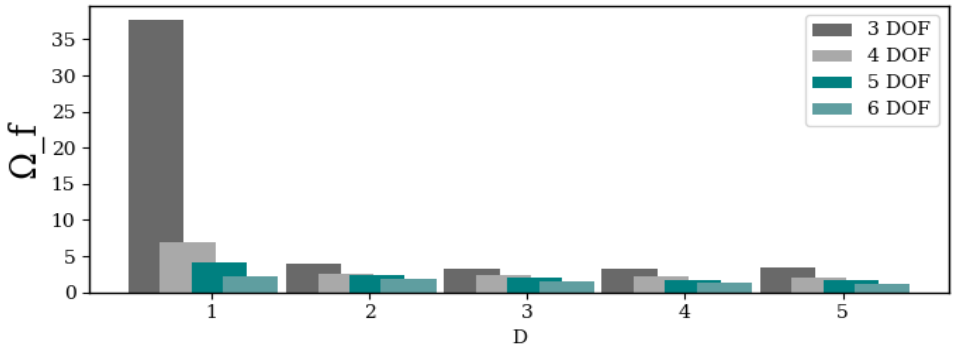
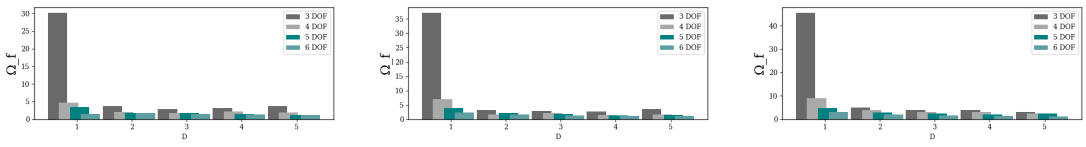


Figure 5.17: Mean valid agent of iteration for all scenarios . $1 \leq D < 6$ **,** $n = 3, 4, 5, 6$



(a) (b) (c)

Figure 5.18: Mean valid agent of iteration . $1 \leq D < 6$ **,** $n = 3, 4, 5, 6$ **for scenario *I* scenario *II* and scenario *III*,(a),(b),(c) respectively.**

Table 5.9: Improvement from PSO in percentage (%) and valid agent found in Ω_f result for $D = 2, 3, 4, 5$ for all n all scenarios. Blue marks improvement compared to the classic PSO.

		Valid Agents improvment (%)				Valid Agents score				
		D				D				
n	scenario	2	3	4	5	1	2	3	4	5
3	*I*	-87.62	-90.49	-89.70	-87.41	30.19	3.74	2.87	3.11	3.80
	II	-91.34	-92.28	-92.76	-90.60	37.17	3.22	2.87	2.69	3.49
	III	-88.89	-91.53	-91.25	-93.33	45.55	5.06	3.86	3.99	3.04
	Mean Improvement(%)	-89.28	-91.43	-91.24	-90.45	37.64	4.01	3.20	3.26	3.44
4	*I*	-56.62	-63.88	-54.89	-60.52	4.73	2.05	1.71	2.14	1.87
	II	-75.76	-69.25	-77.47	-75.14	6.97	1.69	2.14	1.57	1.73
	III	-57.24	-65.08	-64.85	-72.18	9.10	3.89	3.18	3.20	2.53
	Mean Improvement(%)	-63.20	-66.07	-65.74	-69.28	6.94	2.55	2.34	2.30	2.04
5	*I*	-48.53	-50.98	-56.57	-67.10	3.53	1.81	1.73	1.53	1.16
	II	-43.90	-52.04	-66.02	-63.32	3.98	2.23	1.91	1.35	1.46
	III	-39.24	-50.63	-59.68	-50.96	4.88	2.96	2.41	1.97	2.39
	Mean Improvement(%)	-46.21	-51.51	-61.30	-65.21	3.75	2.02	1.82	1.44	1.31
6	*I*	16.44	1.72	-11.51	-14.09	1.47	1.72	1.50	1.30	1.27
	II	-25.55	-43.58	-47.41	-50.45	2.34	1.75	1.32	1.23	1.16
	III	-32.57	-47.52	-51.15	-59.19	3.09	2.08	1.62	1.51	1.26
	Mean Improvement(%)	-13.89	-29.80	-36.69	-41.24	2.30	1.85	1.48	1.35	1.23

5.4.4 Normalized Computation Effort Index (NCEI)

In section 5.4.3, we saw that the RA-PSO converges faster with lower number of valid particles in Ω_f. In order to wisely choose D, we take in consideration both the improvement in the search and the earlier convergence. Therefore, we define a new parameter Θ denoting Normalized Computation Effort Index (NCEI). This parameter is the multiplication between the number of iterations M and the number of valid particles found in Ω_f. Figure 5.19 presents the mean results for all scenarios together. Figure 5.20 presents the results for each scenario separately. The improvement from the PSO is presented in Table 5.10.

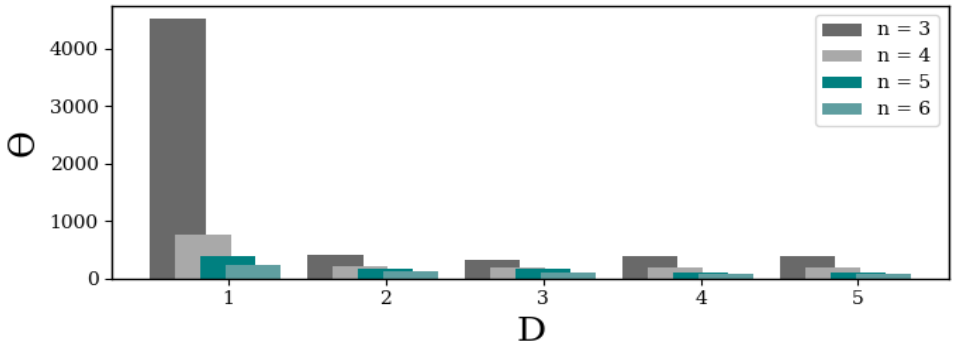
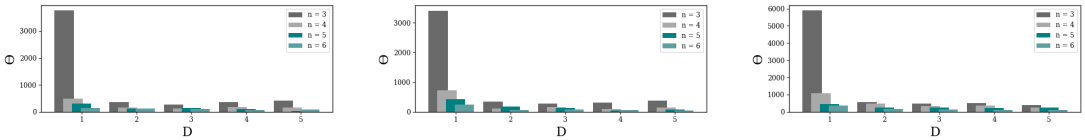


Figure 5.19: Normalized Computation Effort Index for all scenarios . $1 \leq D < 6$ **,** $n = 3, 4, 5, 6$



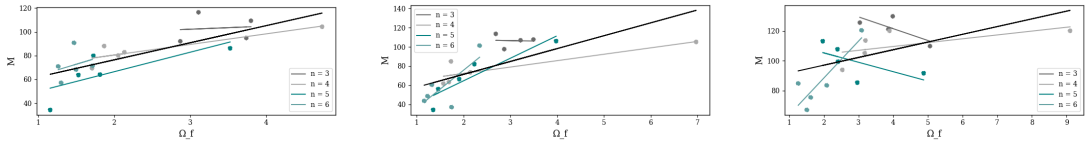
(a) (b) (c)

Figure 5.20: Normalized Computation Effort Index . $1 \leq D < 6$ **,** $n = 3, 4, 5, 6$ **for scenario *I* scenario *II* and scenario *III*,(a),(b),(c) respectively.**

With regards to the NCEI, it is clearly shown that the RA-PSO is better for all n and for all scenarios. The improvement is lower for larger n. We speculate that the reason is due to the large difference in Ω_f from one n to another. Moreover we found a correlation between the number of iterations and the mean valid agents found in Ω_f with very good accuracy. As mentioned before, the classic PSO for $n = 3$ behaves differently from all the other n and we do not consider it in this analysis. The correlation and the statistic results are presented in Figure 5.21 and Table 5.11.

Table 5.10: Mean number of iterations multiplied by the mean number of valid agent, i.e., Θ. Percentage (%) and result for $1 < D < 6$ for all n all scenarios compared to classic PSO

		Θ improvement (%) from PSO				Θ				
		D				D				
n	scenario	2	3	4	5	1	2	3	4	5
3	*I*	-90.58	-92.98	-90.38	-88.96	3763.90	354.57	264.29	362.21	415.49
	II	-89.90	-91.78	-91.04	-88.96	3408.08	344.15	280.25	305.51	376.38
	III	-90.56	-92.06	-91.22	-93.53	5894.00	556.19	467.74	517.65	381.53
	Mean Improvement(%)	-90.35	-92.27	-90.88	-90.48	4355.33	418.30	337.43	395.13	391.13
4	*I*	-66.70	-75.99	-64.18	-66.67	493.84	164.47	118.58	176.92	164.58
	II	-85.39	-78.53	-86.78	-79.94	732.73	107.06	157.34	96.88	146.99
	III	-57.24	-69.43	-66.75	-78.24	1093.46	467.60	334.29	363.53	237.89
	Mean Improvement(%)	-69.77	-74.65	-72.57	-74.95	773.34	246.38	203.40	212.44	183.15
5	*I*	-61.75	-54.58	-67.91	-86.92	303.81	116.21	137.98	97.49	39.73
	II	-56.62	-69.98	-89.01	-80.64	421.96	183.07	126.66	46.39	81.69
	III	-43.40	-46.47	-50.26	-42.38	447.51	253.29	239.54	222.60	257.88
	Mean Improvement(%)	-53.92	-57.01	-69.06	-69.98	391.09	184.19	168.06	122.16	126.43
6	*I*	-8.38	-23.52	-44.30	-32.88	133.79	122.57	102.32	74.51	89.80
	II	-72.70	-66.31	-74.76	-78.65	237.60	64.87	80.04	59.96	50.73
	III	-53.19	-67.09	-72.75	-71.24	372.05	174.14	122.46	101.39	107.02
	Mean Improvement(%)	-44.76	-52.31	-63.94	-60.92	247.81	120.53	101.61	78.62	82.51



(a) (b) (c)

Figure 5.21: Correlation for Iteration by Valid Agent . $1 < D < 6$, $n = 3, 4, 5, 6$ for scenario *I* scenario *II* and scenario *III*,(a),(b),(c) respectively.

Table 5.11: Correlation coefficient and result p-value for Iteration Valid Agent Correlation for all n all scenarios

Scenario	*I*		*II*		*III*	
n	correlation coefficient	p-value	correlation coefficient	p-value	correlation coefficient	p-value
3	0.10	0.90	-0.05	0.95	-0.76	0.24
4	0.88	0.05	0.87	0.06	0.62	0.27
5	0.75	0.15	0.92	0.02	-0.64	0.25
6	0.28	0.64	0.76	0.14	0.88	0.05

There is large difference in performance for various choices of n but not a large difference in the results between the chosen D in each n. To choose D for each n, we take into consideration

also the performance improvement of the RA-PSO. Figure 5.22 presents the improvement in NCEI with regards to the improvement of Φ^* for all n.

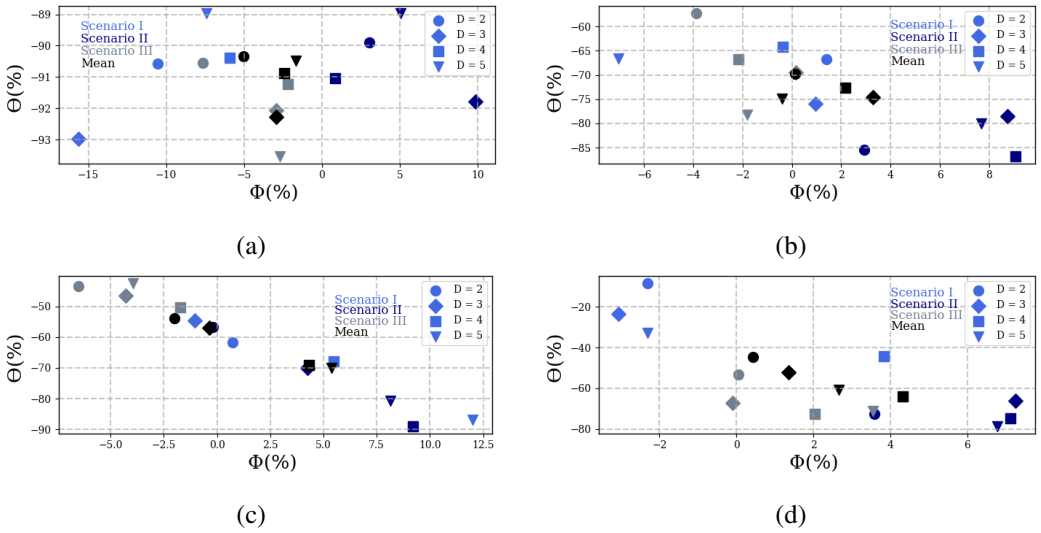


Figure 5.22: Improvement in Θ with regards to improvement in Φ^* for $1 < D < 6$, for $n = 3, 4, 5, 6$ in (a), (b), (c) and (d), respectively.

As mentioned before, we seek for D that provides the best performance both in Ω_f and Θ. From Figure 5.22, we can choose D for each n by its performance in both Θ and Φ^*. For $n = 3$ we choose $D = 2$, although the improvement in Θ is better for $D = 3$ but we prefer the improvement in Φ^*. For $n = 4$, it seems that $D = 5$ presents the best improvement. However, large variation on $D = 5$ between the different scenarios can be seen. $D = 2$ shows lower variation in the results for all the scenarios and the improvement differences in Θ is lower than 5% and in Φ^* is lower than 1%. We choose $D = 2$ for $n = 4$.

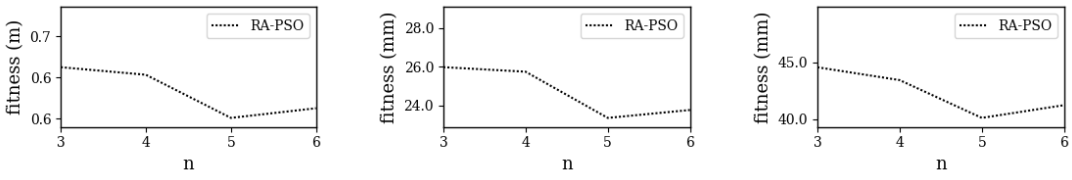
For $n = 5, 6$, $D = 2$ and $D = 3$ show the best results with small variance in the improvement in both Θ and Φ^*. The performance of the RA-PSO with $n = 5$ for $D = 2, 3$ is very low (lower than 2%) while not show difference in the result for Θ. For $n = 5$, we choose $D = 2$. Although, when $n = 6$ the performance in Φ^* is worse in 2%. For $n = 6$ we choose $D = 2$ and give larger significance to the Φ^*.

5.5 Near-optimal robot configuration for all scenarios

This section presents a near-optimal robot configuration for all scenarios following the results of the previous sections. We compare between robots with different n with $D = 2$ so that we can choose the near-optimal robot to perform the tested tasks.

Figure 5.23 presents the results for scenario *I* for all the objective function components. In this scenario, $n = 5$ achieves a better score of 1% than $n = 6$. Furthermore, $n = 4$ gets 9% worse

results compared to $n = 5$ in in the total score Φ^*. For $f(\Phi)$, robots with $n = 4$ track the path better by $2(mm)$ with $n = 5$ and by $5(mm)$ in $E(\Phi)$. In conclusion for scenario *I*, we prefer to choose a robot with $n = 4$ over a robot with $n = 5$. The difference between $n = 4$ to $n = 3$ in scenario *I* is lower than 1% in all testes parameters. Taking all this in consideration for scenario *I*, we choose a robot with $n = 3$. Figure 5.24 presents the chosen robot from that score the best in Φ^* from all the 30 repetitions.



(a) (b) (c)

Figure 5.23: Fitness (a) Φ^*, (b) $f(\Phi)$ and (c) $E(\Phi)$ by $n = 3, 4, 5, 6$ for scenario *I* and $D = 2$.

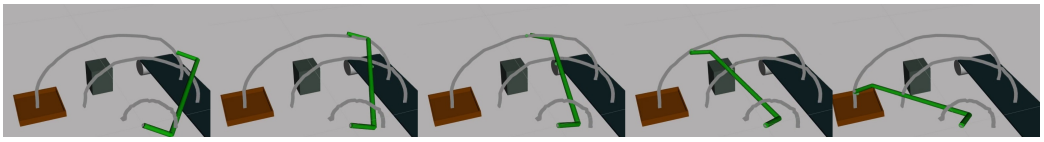
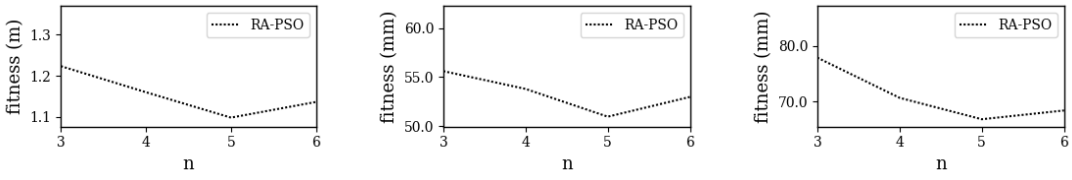


Figure 5.24: Proposed robot configuration with $n = 3$ for scenario *I*.

Figure 5.25 presents the mean results over 30 repetitions for all n with $D = 2$. It seems that the ability to track the path increases with regards to n. A robot with higher n can track the path better. The difference between a robot with $n = 5$ to a robot with $n = 6$ is lower than 3% in $f(\Phi)$ and lower by 2% in $E(\Phi)$. Due to the difficulty of this path scenario, we propose the best configuration to be with $n = 5$. In this scenario, we also want the robot to track the orientation of the hand movement. This, therefore, adds complexity to the robot to track $\mathcal{P}$. Figure 5.26 presents the chosen robot configuration out of 30 repetitions to perform scenario *II* path.



(a) (b) (c)

Figure 5.25: Fitness (a) Φ^*, (b) $f(\Phi)$ and (c) $E(\Phi)$ by $n = 3, 4, 5, 6$ for scenario *II* and $D = 2$.

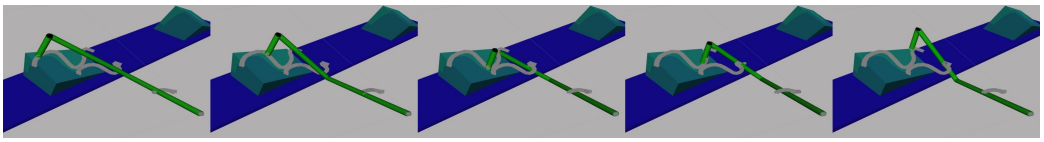
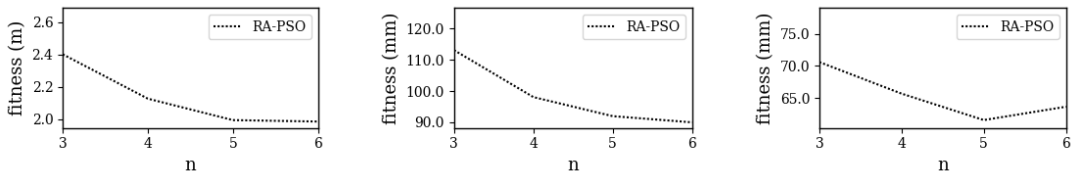


Figure 5.26: Proposed robot configuration with $n = 3$ for scenario *II*.

Figure 5.27 presents the results for all tested parameters with $D = 2$ with regard to n for scenario *III*. Clearly, the ability to track the path increases with n. Although the difference between $n = 5$ to $n = 6$ is 2%, it means that $n = 6$ is able to track the path better by $2(mm)$. For $E(\Phi)$, the difference between $n = 5$ to $n = 6$ is also $2(mm)$. For this scenario, there is no added value for $n = 6$ over $n = 5$. The difference between $n = 5$ to $n = 4$ is more significant: 7%, 6.6% and 6.6% for Φ^*, $f(\Phi)$ and $E(\Phi)$, respectively. This scenario represents the movement in a cluttered environment and a robots ability to track $\mathcal{P}$. Hence, keeping a low $E(\Phi)$ is very important in order to perform the dilution task and move inside the tree canopy. Therefore, we choose $n = 5$ for this scenario as seen in Figure 5.28. Finally, all robot configuration results and DH parameters are summarized in Table 5.12.



(a) (b) (c)

Figure 5.27: Fitness (a) Φ^*, (b) $f(\Phi)$ and (c) $E(\Phi)$ by $n = 3, 4, 5, 6$ for scenario *III* and $D = 2$.

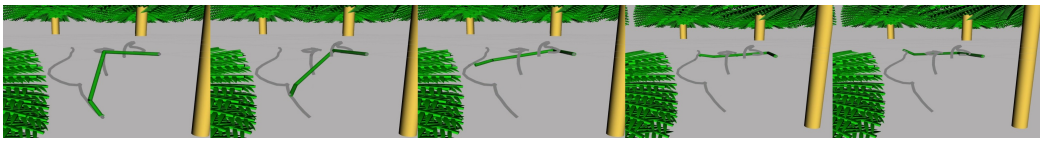


Figure 5.28: Proposed robot configuration with $n = 3$ for scenario *III*.

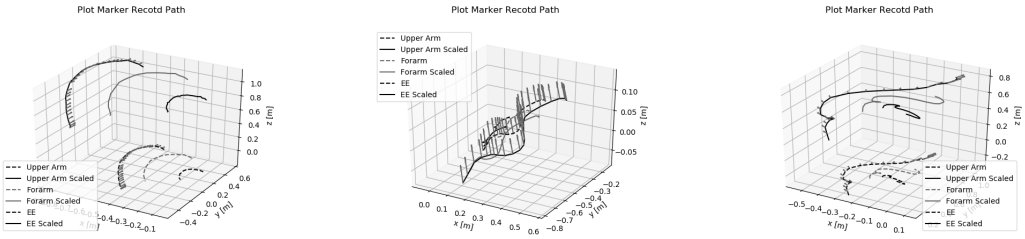
5.6 Robot arm demonstration

This section demonstrates how any off-the-shelf robot can perform the planned path to its best ability and how the path can be fitted to the robot's dimensions. We use the 7-DOF Kinova gen3 arm for this demonstration. As mentioned in section 3, the path $\mathcal{P}$ can be scaled to fit the desired robot length and environment properties. The total length of the Kinova gen3 is $1,187$ (mm)

Table 5.12: Description of the optimal robotic arms

Sc.	n	Φ^*	$f(\Phi)$	$E(\Phi)$	i	α_i	a_i	d_i
I	3	0.43 m	17.59 mm	33.23 mm	0	0.515	0	0
					1	-1.57	0.11	0
					2	-0.27	0.5	0
					EE	-1.57	0	0.1
II	5	0.83 m	35.83 mm	58.32 mm	0	-1.66	0	0
					1	-0.10	0.09	0.31
					2	0.86	0	0
					3	-0.20	0.28	0
					4	0.66	0	0
					EE	-0.2	0	0.1
III	5	1.63 m	72.91 mm	53.79 mm	0	0	0	0
					1	-1.57	0.27	0
					2	-1.10	0	0
					3	00.49	0.33	0
					4	1.51	0	0
					EE	1.57	0	0.1

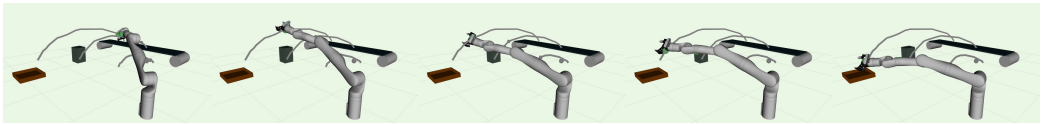
while the arm length of the human demonstrator is 700 (mm). Therefore, the recorded path is scaled up by 1.6 in order to determinate the use of $f(\Phi)$ on the Kinova robot. The scaled path and placement of the Kinova robot for all scenarios is presented in Figure 5.29. Kinova Forword kinematics parameter were taken from the official Kinova specification manual.



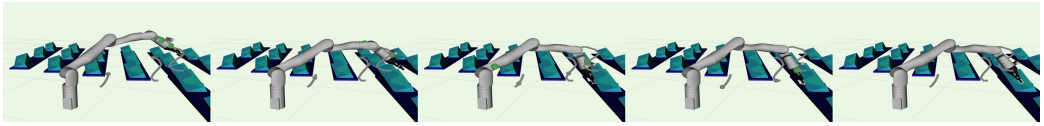
(a) (b) (c)

Figure 5.29: Path $\mathcal{P}$ scaled up by 1.6 for scenarios (a) *I*, (b) *II* and (c) *III*.

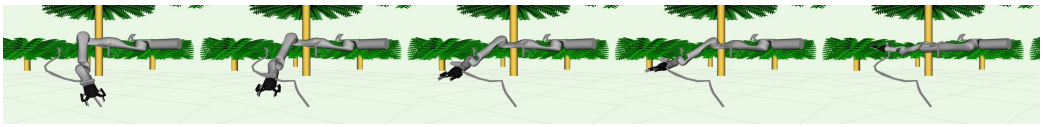
Finlay, we solve equation (3.7) for the Kinova configuration. Figure 5.31 presents snap-shots of the Kinova robot performing all scaled up scenarios in Gazebo-ROS simulation. Table **??** presents the fitness scores of the Kinova performing $\mathcal{P}$ for all the tested scenarios. Kinova gen3 arm is a 7-DOF robot which means that our algorithm is not limited for only up to 6 DOF robot configurations.



(a)



(b)



(c)

Figure 5.30: Kinova gen3 perform the optimal paths for scenarios (a) *I*, (b) *II* and (c) *III*.

Table 5.13: Kinova performance scores for all scenarios.

	$\Phi^*(m)$	$f(\Phi)(mm)$	$E(\Phi)(m)$
scenario *I*	1.56	76.6	0.08
scenario *II*	2.37	87.8	0.21
scenario *III*	5.50	283.3	0.12

Moreover, we test the Kinova performing scenario *III* in a mock-up tree built in the laboratory. While changing the start point of the robot. The results show that the Kinova can interact with the tree model leaves in order to perform the thinning task successfully.

Figure 5.31: Kinova gen3 roll-outs of the optimal path for scenario *III* with a tree mock-up on different approach points.

6 Conclusions

In this work, we present an approach to find a near-optimal robot configuration and placement in the world to track a recorded path demonstrated by a human expert. We find that best algorithm to solve multi-point Inverse Kinematics in terms of computation time is the PSO algorithms. However, the differences between the meta-heuristic methods for this problem are minor and a different algorithm will have the same results in this optimization problem.

Furthermore, we find that the best algorithm to solve the robot configuration optimization problem is the classic PSO algorithm, and suggest a new approach. The RA-PSO, based on the PSO algorithm, improves computation time and effort. Our new approach can save significant hardware resources and time while keeping the same results as the classic method. Furthermore, we have defined a new evaluation index termed *Normalized computation effort index* that combines both the convergence iteration and valid particles.

This method was tested on three test cases. Classic pick and place of an object from a conveyor while avoiding an obstacle; track a welding path while keeping the EE orientation; and the dilution of a palm trees. The palm tree is a cluttered environment that requires the entire arm to interact with the obstacles while preforming the path. In order to establish the palm tree canopy as an cluttered environment with static and elastic obstacle, we modeled the mechanical behavior of the palm tree leaf.

Finally, for each test case we compare the different DOF robots performance and suggest a near-optimal robot configuration to preform each of the tasks. Moreover, to test the movement of the robot in the cluttered environment, we rolled-out the recorded path with the Kinova robot in a scaled model of a palm tree canopy. We placed the robot in 4 different positions associated with the tree and the robot moved and pushed the tree leaves in order to preform the task. Results show that this method is not suitable to find both the robot orientation and position in high accuracy. If the weight of the orientation is higher than 0.3, the robot cannot sufficiently track the position accurately in order to avoid obstacles or to interact with the environment.

Future work could focus on timing the weights along the path to mark what is more important according to the task: to keep the EE orientation or to keep the link of the robot in a specific pose. Moreover, a new path planner can be modified to teach the robot to use its all body links to interact with the environment.Furthermore, the dynamic leaf model can be used to determine

torque magnitudes that are required from the actuators in order to push the tree leaf. Moreover, we can recorded several paths for the same task and suggest the near- optimal robot to perform all paths.

References

[1] M. Cefalo, G. Oriolo, and M. Vendittelli, "Planning safe cyclic motions under repetitive task constraints," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 3807–3812.

[2] H. M. Do, C. Park, and J. H. Kyung, "Dual arm robot for packaging and assembling of it products," in *IEEE International Conference on Automation Science and Engineering*, 2012, pp. 1067–1070.

[3] M. Beschi, S. Mutti, G. Nicola, M. Faroni, P. Magnoni, E. Villagrossi, and N. Pedrocchi, "Optimal robot motion planning of redundant robots in machining and additive manufacturing applications," *Electronics*, vol. 8, p. 1437, Dec 2019.

[4] P. Chutima, "A comprehensive review of robotic assembly line balancing problem," *J. of Intelligent Manuf.*, vol. 8, pp. 1572–8145, 2020.

[5] V. Bloch, A. Degani, and A. Bechar, "A methodology of orchard architecture design for an optimal harvesting robot," *Biosystems Engineering*, vol. 166, pp. 126–137, 2018.

[6] M. Perrollaz, S. Khorbotly, A. Cool, J. Yoder, and E. Baumgartner, "Teachless teach-repeat: Toward vision-based programming of industrial robots," in *IEEE Inter. Conf. on Rob. & Auto.*, 2012, pp. 409–414.

[7] W. Feller, *An Introduction to Probability Theory and Its Applications, Vol. 1, 3rd Edition*, 1968.

[8] L. Kavraki, M. Kolountzakis, and J.-C. Latombe, "Analysis of probabilistic roadmaps for path planning," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 1, pp. 166–171, 1998.

[9] T. Horsch, F. Schwarz, and H. Tolle, "Motion planning with many degrees of freedom-random reflections at c-space obstacles," in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, 1994, pp. 3318–3323 vol.4.

[10] L. E. Kavraki, J.-C. Latombe, R. Motwani, and P. Raghavan, "Randomized query processing in robot path planning," *Journal of Computer and System Sciences*, vol. 57, no. 1, pp. 50–60, 1998. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0022000098915781

[11] M. R. Dogar and S. S. Srinivasa, "Push-grasping with dexterous hands: Mechanics and a method," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 2123–2130.

[12] J. Denavit and R. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," *Journal of Applied Mechanics*, vol. 77, pp. 215–221, 1995.

[13] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *International Conference on Neural Networks*, vol. 4, 1995, pp. 1942–1948.

[14] Shicai Shi, Xiaohui Gao, Zongwu Xie, Fenglei Ni, Hong Liu, E. Kraemer, G. Hirzinger, and S. C. Shi, "Development of reconfigurable space robot arm," in *International Symposium on Systems and Control in Aerospace and Astronautics*, 2006, pp. 6 pp.–143.

[15] R. Vijaykumar, K. Waldron, and M. Tsai, "Geometric optimization of serial chain manipulator structures for working volume and dexterity," *Int. Journal of Robotics Research*, vol. 5, no. 2, pp. 91–103, 1986.

[16] T. S. Sarosh H. Patel, "Optimal design of threelink planar manipulators using grashof's criterion," *IGI Global*, 2012.

[17] A. Zeiaee, R. Soltani-Zarrin, R. Langari, and R. Tafreshi, "Kinematic design optimization of an eight degree-of-freedom upper-limb exoskeleton," *Robotica*, vol. 37, no. 12, p. 2073–2086, 2019.

[18] W. S. You, Y. Lee, G. Kang, H. Oh, J. Seo, and H. Choi, "Kinematic design optimization for anthropomorphic robot hand based on interactivity of fingers," *Intelligent Service Robotics*, pp. 1–12, 04 2019.

[19] M. Ceccarelli and C. Lanni, "A multi-objective optimum design of general 3r manipulators for prescribed workspace limits," *Mechanism and Machine Theory*, vol. 39, no. 2, pp. 119–132, 2004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094114X03001095

[20] G. C. M. and G. M., "The Synthesis of Manipulators with Prescribed Workspace," *Journal of Mechanical Design*, vol. 113, no. 4, pp. 451–455, 12 1991.

[21] B. M. J., "A New Method of Constrained Optimization and a Comparison With Other Methods," *The Computer Journal*, vol. 8, no. 1, pp. 42–52, 04 1965. [Online]. Available: https://doi.org/10.1093/comjnl/8.1.42

[22] O. Ma and J. Angeles, "Optimum design of manipulators under dynamic isotropy conditions," pp. 470–475 vol.1, 1993.

[23] M. O. and A. J., "The concept of dynamic isotropy and its applications to inverse kinematics and trajectory planning," pp. 481–486 vol.1, 1990.

[24] W. Jun and W. L. Y. Zheng, "A new method for optimum design of parallel manipulator based on kinematics and dynamics," *Nonlinear Dynamics*, vol. 61, pp. 717–727, 09 2010.

[25] K. Abdel-Malek and W. Yu, "Design optimization of robot grippers using teaching-learning-based optimization algorithm," *International Journal of Robotics and Automation*, 2004.

[26] J.-Y. Park, P.-H. Chang, and J.-Y. Yang, "Task-oriented design of robot kinematics using the grid method," *Advanced Robotics*, vol. 17, no. 9, pp. 879–907, 2003. [Online]. Available: https://doi.org/10.1163/156855303770558679

[27] G. Yang and I.-M. Chen, "Task-based optimization of modular robot configurations: minimized degree-of-freedom approach," *Mechanism and Machine Theory*, vol. 35, no. 4, pp. 517–540, 2000. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094114X9900021X

[28] M. J. H. Lum, J. Rosen, M. N. Sinanan, and Hannaford B, "Kinematic optimization of a spherical mechanism for a minimally invasive surgical robot," in *IEEE Int. Conf. on Robotics and Automation*, vol. 1, 2004, pp. 829–834.

[29] A. Kuntz, C. Bowen, C. Baykal, A. W. Mahoney, P. L. Anderson, F. Maldonado, R. J. Webster, and R. Alterovitz, "Kinematic design optimization of a parallel surgical robot to maximize anatomical visibility via motion planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 926–933.

[30] A. W. Mahoney, P. L. Anderson, P. J. Swaney, F. Maldonado, and R. J. Webster, "Reconfigurable parallel continuum robots for incisionless surgery," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 4330–4336.

[31] J. Burgner, H. B. Gilbert, and R. J. Webster, "On the computational design of concentric tube robots: Incorporating volume-based objectives," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 1193–1198.

[32] M. Feng, X. Jin, W. Tong, X. Guo, J. Zhao, and Y. Fu, "Pose optimization and port placement for robot-assisted minimally invasive surgery in cholecystectomy," *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 13, no. 4, p. e1810, 2017, e1810 RCS-16-0140.R2. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rcs.1810

[33] S. Patel and T. Sobh, "Task based synthesis of serial manipulators," *Journal of Advanced Research*, vol. 6, no. 3, pp. 479 – 492, 2015.

[34] D. P. Garg and M. Kumar, "Optimization techniques applied to multiple manipulators for path planning and torque minimization," *Engineering Applications of Artificial Intelligence*, vol. 15, no. 3, pp. 241–252, 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197602000672

[35] M. D.C. and M. M.K., "Planning of robot trajectories with genetic algorithms," pp. 223–228, 1999.

[36] E. Solteiro Pires and J. Tenreiro Machado, "A trajectory planner for manipulators using genetic algorithms," pp. 163–168, 1999.

[37] A. J. Scarfe, R. C. Flemmer, H. H. Bakker, and C. L. Flemmer, "Development of an autonomous kiwifruit picking robot," in *2009 4th International Conference on Autonomous Robots and Agents*, 2009, pp. 380–384. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/4804023

[38] E. Van Henten, D. Van't Slot, C. Hol, and L. Van Willigenburg, "Optimal manipulator design for a cucumber harvesting robot," *Computers and Electronics in Agriculture*, vol. 65, no. 2, pp. 247–257, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169908002238

[39] N. Vahrenkamp, T. Asfour, G. Metta, G. Sandini, and R. Dillmann, "Manipulability analysis," in *IEEE-RAS International Conference on Humanoid Robots*, 2012, pp. 568–573.

[40] L. Stocco, S. E. Salcudean, and F. Sassani, "Fast constrained global minimax optimization of robot parameters," *Robotica*, vol. 16, no. 6, p. 595–605, 1998.

[41] J. Rastegar and B. Fardanesh, "Manipulation workspace analysis using the monte carlo method," *Mechanism and Machine Theory*, vol. 25, no. 2, pp. 233 – 239, 1990.

[42] B. Paden and S. Sastry, "Optimal kinematic design of 6r manipulators," *Int. Journal of Robotics Research*, vol. 7, no. 2, pp. 43–61, 1988.

[43] M. Stock and K. Miller, "Optimal kinematic design of spatial parallel manipulators: Application to linear delta robot," *Journal of Mechanical Design*, vol. 125, 06 2003.

[44] Z. B. S. Kucuk, "Robot workspace optimization based on a novel local and global performance indices," *IEEE ISIE*, pp. 20–23, 2005.

[45] C. Giladi and A. Sintov, "Manifold learning for efficient gravitational search algorithm," *Information Sciences*, vol. 517, pp. 18 – 36, 2020.

[46] S. Khatami and F. Sassani, "Isotropic design optimization of robotic manipulators using a genetic algorithm method," in *IEEE Internatinal Symposium on Intelligent Control*, 2002, pp. 562–567.

[47] J. Bryson, X. Jin, and S. Agrawal, "Optimal design of cable-driven manipulators using particle swarm optimization," *ASME. J. Mechanisms Robotics*, vol. 8, no. 4, 2016.

[48] L. Smith, N. Dhawan, M. Zhang, P. Abbeel, and S. Levine, "Avid: Learning multi-stage tasks via pixel-level translation of human videos," in *Robotics: Science and Systems (RSS)*, 2020.

[49] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas, and N. Heess, "Reinforcement and imitation learning for diverse visuomotor skills," in *Proceedings of Robotics: Science and Systems*, 2018.

[50] N. García, J. Rosell, and R. Suárez, "Motion planning by demonstration with human-likeness evaluation for dual-arm robots," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 11, pp. 2298–2307, 2019.

[51] A. Kapusta and C. C. Kemp, "Optimization of robot configurations for assistive tasks," *Georgia Tech Library*, 2016.

[52] A. Perez-Gracia and J. M. McCarthy, "Kinematic synthesis of spatial serial chains using clifford algebra exponentials," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 220, no. 7, pp. 953–968, 2006.

[53] S. Shirafuji and J. Ota, "Kinematic synthesis of a serial robotic manipulator by using generalized differential inverse kinematics," *IEEE Transactions on Robotics*, vol. 35, no. 4, pp. 1047–1054, 2019.

[54] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*, 1st ed. CRC Press, Mar. 1994.

[55] T. Dokeroglu, E. Sevinc, T. Kucukyilmaz, and A. Cosar, "A survey on new generation metaheuristic algorithms," *Computers & Industrial Engineering*, vol. 137, p. 106040, 2019.

תקציר

זרועות רובוטיות הן הבסיס לאוטומציה מודרנית לייצור. הן ממוקמות בקווי ייצור ומבצעות את רוב המשימות כגון הרכבה, עיבוד שבבי, צביעה, ריתוך ואריזה. עם זאת, רובוטים אלו בעלי יכולת גבוהה בדרך כלל מורדים למשימות פשוטות וחזרתיות כגון איסוף ממסוע ומיקום במקום יעודי אחר או ריתוך של אותו מסלול. מצד שני, עיצוב רובוט אופטימלי עבור משימה ספציפית אחת גוזל משאבים גדולים של זמן הנדסי ועלויות. יתר על כן, השיטות הנפוצות לתכנון מסלול מחפשות נתיבים ללא התנגשות עבור הזרועות הרובוטיות. עם זאת, לא סביר שימצאו מסלול בסביבות עמוסות בהן יש לפנות חפצים על מנת להגיע למטרה. כמו כן, זרועות רובוטיות בדרך כלל יוצרות אינטראקציה עם הסביבה אך ורק באמצעות אביזר הקצה שלהן.

עבודה זו מחפשת את תצורת הרובוט אופטימלית לביצוע משימה מוגדרת המבוססת על חיקוי מדגים אנושי. יתרה מכך, שיטה זו מחפשת את משתנה קינמטיקת הרובוט ואת המיקום של רובוט. רובוט אופטימלי הוא זה שמשלב את מספר דרגות החופש המינימלי ומספק את הדיוק הטוב ביותר במהלך ביצוע המשימה. בנוסף, הקינמטיקה האופטימלית ניתנת עם האוטומציה הנדרשת להשלמת המשימה. הנתיב מחושב בשיטה מטה-יוריסטית כדי למצוא את ערכי המנועים של הרובוט לביצוע המשימה. גישה זו לוקחת בחשבון את המכלול זרוע-רובוט (מפרקים וקישורים), לביצוע משימות בסביבה עמוסה או כדי להימנע ממכשולים. השיטה המוצעת יכולה לשמש גם לתכנון מסלול לרובוט קיים. אנו מספקים ניתוח השוואתי כדי לזהות את האלגוריתם המתאים ביותר לפתור את בעיית אופטימיזציה זו. שיטות ידועות שונות נבדקו והשוו. יתר על כן וכדי להתגבר על מרחב חיפוש מאוד לא לינארי ולא רציף של הבעיה, מוצע אלגוריתם חדש. השיטה החדשה יכולה למצוא את עיצוב של רובוט אשר מקבל את אותה תוצאה לפי הפרמטרים שהוגדרו כמו השיטה הסטנדרטית ואף עם מאמץ חישוב נמוך יותר.

כדי לבדוק ולבסס את השיטה שלנו אנו בוחנים שלושה מקרים שונים. הראשון הוא איסוף חפץ ממסוע והנחתו בקופסא תוך הימנעות ממכשול, התרחיש השני הוא רובוט ביצוע משימת ריתוך ובו אנו דורשים לעקוב אחר המיקום והכיוון של תנועת יד האדם. התרחיש האחרון הוא רובוט שנע בתוך צמרת עץ דקל על מנת לבצע את משימת דילול התמרים. צמרת עץ הדקל מייצגת סביבה עמוסה במכשולים. על מנת לבצע את המשימה על הרובוט להיות באינטראקציה עם העלים של העץ עם כל חלקיו ולא רק עם אביזר הקצה.

אוניברסיטת תל אביב

הפקולטה להנדסה ע"ש איבי ואלדר פליישמן

בית הספר לתארים מתקדמים ע"ש זנדמן סליינר

אופטימיזצית קונפיגורציה של זרוע רובוטית המבצעת משימות בסביבות צפופות מכשולים

חיבור זה הוגש כעבודת מחקר לקראת התואר "מוסמך אוניברסיטה" בהנדסה מכנית

על ידי

ענבר מאיר

העבודה נעשתה בבית הספר להנדסה מכנית בהנחיית ד"ר אבישי סינטוב מטעם המחלקה להנדסה מכנית ופרופ' אביטל בכר המכון להנדסה חקלאית מרכז ולקני

אב תשפ"ב