Methods for Simultaneous Orchard and Harvesting Robot Design

Research Thesis
in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

Victor Bloch

Submitted to the Senate of the Technion – Israel Institute of Technology
Sivan, 5777 Haifa June, 2017
The Research Thesis Was Done under the Supervision of Assist. Prof. Amir Degani in the Faculty of Civil and Environmental Engineering and Dr. Avital Bechar in the Institute of Agricultural Engineering, ARO, Volcani Center
Publication List

Journal Paper


Book Chapters and Conference Papers


# Table of Contents

List of Figures .......................................................................................................................... 1
Abstract .................................................................................................................................. 8
List of Symbols and Acronyms ................................................................................................. 10
1. Introduction .......................................................................................................................... 12
   1.1. Problem description ........................................................................................................ 12
   1.2. Objectives ..................................................................................................................... 13
   1.3. Significance ................................................................................................................... 14
   1.4. Innovation .................................................................................................................... 15
   1.5. Contribution .................................................................................................................. 15
   1.6. Thesis structure ............................................................................................................ 16
2. Literature Review .................................................................................................................. 17
   2.1. Robot design ................................................................................................................. 17
   2.2. Robot design in agriculture .......................................................................................... 18
   2.3. Tree modeling and design ............................................................................................ 20
3. Methodology ........................................................................................................................ 23
   3.1. Field data collection and modeling of plants .................................................................. 23
   3.2. Simulation of robot performance .................................................................................. 25
   3.3. Optimization of robot and environment structures ....................................................... 27
   3.4. Statistical analysis of the geometrical features of the environment .............................. 27
   3.5. Field experiment .......................................................................................................... 27
   3.6. Approach and assumptions .......................................................................................... 27
4. Task-Based Robot Optimization ........................................................................................... 37
   4.1. Robot performance cost function .................................................................................. 37
   4.2. Optimization parameters ............................................................................................... 40
   4.3. Optimization constraint ............................................................................................... 41
   4.4. Sub-problems of the robot optimization ....................................................................... 41
   4.5. Methods for robot kinematics optimization ................................................................ 43
   4.6. Influence of the optimization constraint on the cost function ..................................... 45
10. Discussion ........................................................................................................................................... 141

11. Conclusions and Future Work ........................................................................................................... 144

Appendix A. Environment Library ........................................................................................................... 147

Appendix B. Analysis of the Characterization ....................................................................................... 158

Appendix C. Statistical Analysis of Actual Environments .................................................................... 168

Appendix D. Initial Field Experiment ................................................................................................... 175

Bibliography ........................................................................................................................................... 181
List of Figures

Fig. 3.1  The digitizer, a mechanical measuring device, is a passive robotic arm with RRRR structure.  
Fig. 3.3  Views of agricultural environments: (a) row view and (b) side view of an apple tree shaped by the Central Leader method.  *The coordinate system is shown by arrows. The location of the fruit is defined as shown in (c).*  
Fig. 3.4  Agricultural tasks with obvious structural differences: orange, pear and mango picking.  
Fig. 4.1  Robotic arms with 3 degrees of freedom of the robotic arm and different kinematic structures: PPP (a), RRP (b) and RRR (c).  *The arms are located on a platform vehicle, which is not considered part of the arm’s kinematics.*  
Fig. 4.2  Surface describing the cost function of a robot depending on variables a2 and d1, $F(a_2,d_1)$.  *The front (a), side (b), top (c) and isometric (d) views are shown. The colors of the surface represent the surface height proportional to the F value. The local minima are presented by red asterisks.*  
Fig. 4.3  Robots with structures PPP (a), RRP (b) and RRR (c) optimal for fruit picking on the model Tall Spindle apple tree with two optimal robot locations.  
Fig. 4.4  Robots with the structures PPP (a), RRP (b) and RRR (c) optimal for fruit picking on the model Y-trellis apple tree with four optimal robot locations.  
Fig. 4.5  Robots with the structures PPP (a), RRP (b) and RRR (c) optimal for fruit picking on the model Central Leader apple tree with four optimal robot locations.  
Fig. 4.6  Dependence $F(\alpha_{allow})$ for the RRR robot working on the Central Leader apple tree model: all considered values of $F$ (a), enlargement for $0.4 < F < 1$ (b).  
Fig. 4.7  Model of Y-trellis apple row with five trees is too complicated to find optimal robots (a).  *Non-optimal robot performing the picking task for the Y-trellis apple row (b).*  
Fig. 4.8  Robot kinematics of the FANUC P-250iA/15 (a), FANUC P-250iA/10S (b), KUKA KR 16 L6-2 (c) and Yaskawa Motoman HP20D-A80 (d) models (taken from the robot specifications of FANUC, KUKA and Yaskawa companies).  
Fig. 5.1  Close-to-optimal robot locations achieved by the simplex method with 20 random initial guesses presented by dots in different colors.  *The near-optimal robot locations achieved by the grid search method are shown as asterisks (a). The distribution of the cost-function values for the found solution is presented in the histogram (b).*  
Fig. 5.2  Close-to-optimal robot locations achieved by 20 runs of the genetic algorithm.  *The near-optimal robot locations achieved by the grid search method are shown as asterisks (a). The distribution of the cost-function values for the found solution is presented in the histogram (b).*  
Fig. 5.3  Possible robot locations around a tree distributed concentrically and
represented by dots. *Robot location inside the sectors with open angle of 60° is assumed to be impossible.*

Fig. 5.4 Possible robot locations along the tree row, represented by dots. 63
Fig. 5.5 Division of possible robot locations into equal sectors around the tree. 68
Fig. 5.6 Eight iterations of the algorithm for the optimal location combination search. The possible locations are represented by dots with different colors corresponding to the different sectors. 76

Fig. 6.1 Developmental stages of the environment characterization method. 77
Fig. 6.2 2D RR robotic arm for fruit picking (a). *The robot is modeled by lines representing the robot links with the base at the origin (b). The robot reaches its targets, represented by red dots.* 78
Fig. 6.3 Small clusters of actual targets surrounded by red ellipses are characterized by the mean characteristic fruit, represented by green stars. *The farthest fruit is represented by a black triangle.* 79
Fig. 6.4 Clustering of an environment. The points represent the actual targets, the asterisks represent the mean characteristic targets. *The percentage of actual targets characterized by each mean characteristic target appears beside the asterisk.* 84
Fig. 6.5 The dependence of dP, dF, dL and αuf on Nmean. 84
Fig. 6.6 Optimal robots based on the actual environment (RAE) (solid lines) and on the characteristic environment (RCE) (dashed lines) achieved for characteristic environments (CE) with different numbers of clusters: 1, 3 and 6. *The actual targets constituting the actual environment (AE) are represented as dots, the characteristic targets constituting CE are represented as asterisks. The home configuration is shown by solid continuous lines for RAE and dashed lines for RCE.* 85
Fig. 6.7 Comparison of the performances of optimal robots based on the actual environment (RAE) and on the characteristic environment (RCE) in the actual environment. *The robot base is located far (a) and close (b) to the targets. The targets that are not approached by the RCE are marked by black circles.* 86
Fig. 6.8 The targets that are unreachable by the optimal robot based on the characteristic environment (RCE) in Fig. 6.7 are outside the limits defined by the arcs with the robot base at their center as they pass through the extreme targets. 87
Fig. 6.9 Optimal robots based on the actual environment (RAE) and on a characteristic environment (RCE) with extreme characteristic targets: Nfar = 1, Nnear = 0 in (a), and Nfar = 1, Nnear = 1 in (b). *The actual targets are represented as dots, the farthest extreme characteristic targets are represented as Δ, the nearest extreme characteristic targets are represented as V.* 88
Fig. 6.10 Optimal robots based on the actual environment (RAE) and on a characteristic environment (RCE) with different numbers of mean characteristic targets, Nmean = 1...4, and extreme characteristic targets with Nfar = 1, Nnear = 1. *The actual targets are represented as dots, the mean
characteristic targets as asterisks, the farthest extreme characteristic targets as Δ, the nearest extreme characteristic targets as V.

Fig. 6.11 Optimal robots for α_{allow} = [0, 10%, 20%]. The unpicked fruit are marked by circles.

Fig. 6.12 Dependence of the robot performance cost function F2D on the allowed percentage of unpicked fruit, α_{allow}, for 100 random actual environments (AE). The thin lines represent the dependence for a specific AE. The thick red line represents the average.

Fig. 6.13 Environment characterization with allowed unreachable targets. The farthest unreachable targets are marked by squares. The farthest characteristic target is marked by Δ. The actual reachable targets are bounded by an arc with the center in the robot base location passing through the farthest characteristic target.

Fig. 6.14 Mapping between the target location and the robot cost function for reaching this target. A robot with a specific kinematics and base location is taken and shown in (a). The surface of the cost function depending on the target location F2D(x,y) in the domain 0 ≤ x ≤ 2, 0 ≤ y ≤ 2 is shown in (b). The contours of the surface are compared with the concentric circles with the robot base as center in (c).

Fig. 6.15 Characterization of an environment with α_{allow} = 10% by mean targets and extreme targets achieved by clustering into two (a) and three (b) clusters. The optimal robot based on the actual environment reaching the targets is shown in (c), and the unreached targets are marked by circles.

Fig. 6.16 Extreme farthest (a) and nearest (b) fruit with similar locations but different orientations define different robot configurations. P_{target}, V_{target} and P_{base} are the target location, target orientation vector and robot base location, respectively.

Fig. 6.17 Views of the Tall Spindle apple tree model: with obstacles (a), without obstacles (b). The fruit, presented as lines, are shown in side (c) and front (d) views.

Fig. 6.18 Optimal robot based on the actual environment for Tall Spindle apple tree model with different structures: RRR (a), RRP (b) and PPP (c).

Fig. 6.19 Characteristic environments for the Tall Spindle apple tree's actual environment with 2 clusters. The actual fruit are depicted by dots with different colors corresponding to different clusters. The mean targets of the clusters are depicted as asterisks with the cluster percentage. The farthest and nearest extreme fruit of the clusters are depicted as Δ and V, respectively. The assumed robot base position is depicted by a circle.

Fig. 6.20 Characteristic environments for the Tall Spindle apple tree's actual environment with 3 clusters. All signs are as in Fig. 6.19.

Fig. 6.21 Characteristic environments for the Tall Spindle apple tree's actual environment with 6 clusters. All signs are as in Fig. 6.19.

Fig. 7.1 Parameters of a circular reaching cone. The cone vertex is located at the fruit marked in yellow, the cone axis is defined by the vector \( \mathbf{n} \), the cone opening angle is \( \alpha \), the cone height is \( h \), the surface is defined by three points of
intersection with the obstacles depicted by the asterisks.

Fig. 7.2 Defining the direction of beams starting from a fruit.

Fig. 7.3 Tall Spindle apple tree (left) and Central Leader apple tree (right) models with the fruit taken as examples for the reaching cone calculations. The considered fruit are marked by circles.

Fig. 7.4 Star maps for the considered fruit. Black pixels represent rays which do not reach obstacles. The gray-scale pixels represent rays reaching obstacles with the intensity of the color representing the distance according to the shown scale.

Fig. 7.5 Binary star maps. The grid of 36 uniformly distributed initial guesses, represented by red asterisks (a). The projection of the umbrella base circle, in blue (b), at the beginning of the umbrella opening at one of the initial guesses.

Fig. 7.6 The process of umbrella opening modeled by the increase and movement of the projection of the umbrella base circle. (a) Increasing the base projection until collision with an obstacle. (b) Increasing the base projection touching the obstacle until collision with a second obstacle. (c) Increasing the base projection touching two obstacles until collision with a third obstacle. (d) Collision of the base projection with three obstacles defines the base projection size and location in the map and the umbrella opening angle and direction. The arrow shows the direction of the circle projection movement while it increases.

Fig. 7.7 Views of a reaching cone for fruit 16. The cone vertex is located in the center of the fruit. The cone surface is defined by three points of collision with obstacles: tree stem, trellis wire and one of the tree branches.

Fig. 7.8 Definition of the difference between two cones.

Fig. 7.9 Seven reaching cones represented as the projections of the cones' base circles in the star map of fruit 16 and as cones in the Cartesian coordinate system.

Fig. 7.10 Map of the reaching cone projections for fruit 7 on the Tall Spindle apple model and isometric views of the cones.

Fig. 7.11 Map of the reaching cone projections of fruit 40 in the Central Leader apple model and isometric views of the cones.

Fig. 7.12 Coordinate systems defining the location of a point around the tree model.

Fig. 7.13 Reachability map for the Central Leader apple tree model with fixed distance \( r = 1 \) m. The values of the reachable fruit percentage strongly depend on the location around the tree.

Fig. 7.14 Dependence of the total percentage of reachable fruit on the number of point locations \( N_{\text{loc}} \). The near-optimal locations are presented by the asterisks in (a). The graph of the dependence is shown in (b).

Fig. 8.1 Choosing the points suitable for the robot location among all possible robot locations for the Central Leader apple tree (a,b) and Y-trellis apple tree (c,d). The possible points are represented by dots, and the suitable points are surrounded by circles in (a) and (c). A surface approximating the location suitability is passed through the calculated \( \phi \) values in (b) and (d). The triangles in (a) and (c) represent the robot locations found while considering
all possible robot locations, the squares represent the robot locations found based on the suitable robot locations.

Fig. 8.2 Choosing the points suitable for the robot location among all possible robot locations for a row of Y-trellis apple trees. The possible points are represented by dots, and the suitable points are surrounded by circles (a). A surface approximating the location suitability is passed through the calculated $\phi_i$ values (b). The triangles in (a) represent the robot locations found while considering all possible robot locations, the squares represent the robot locations found based on the suitable robot locations.

Fig. 8.3 Central Leader tree model (a) and its characteristic environment constructed by clustering and reaching cones for the Central Leader model.

Fig. 8.4 Optimal robot based on a characteristic environment for the Y-trellis row.

Fig. 8.5 Optimal robot based on a characteristic environment for the Central Leader row.

Fig. 8.6 Optimal robot based on a characteristic environment for the Nectarine row.

Fig. 9.1 Three actual task environments and their models: apple trees trained by Central Leader (a), Tall Spindle (b) and Y-trellis (c) training systems.

Fig. 9.2 Total fruit picking time for different orchard types, robot types and number of locations depending on the movement time $T_{mov}$.

Fig. 9.3 Three models of trees shaped by the Y-trellis method with different trellis tilt angle: $5^\circ$ (a), $30^\circ$ (b) and $85^\circ$ (c).

Fig. 9.4 Performance cost function of the optimal robots with RRR (a) and RRP (b) structures for Y-trellis tree models. The trellis tilt angles are in the range $[5^\circ..85^\circ]$. The number of robot locations around the tree, $N_{loc}$, is 2, 4, 6 and 8. Starting from $45^\circ$ for the RRP robot, $N_{loc} = 2$, there are no solutions for the optimal robots fulfilling the optimization constraint.

Fig. D.1 The robot in the field experiment. The platform with the robotic arm (a). The robotic arm approaching one of the artificial fruit (b). The angle $\theta_2$ equals $70^\circ$ in the figure.

Fig. D.2 The harvesting system. (a) The Whooshh platform with the vacuum tube for fruit transport and the picking stations for the workers. (b) The robotic arm installed on one of the picking stations of the platform (c) performs the apple harvesting inside an orchard. (d) The robot end effector is a soft gripper.

Fig. D.3 Actual task environment (a) and its model (b).

Fig. D.4 Fruit-picking experiment and its simulation. The actual robotic arm and the tree (a) are compared with their model (b). The models of the robotic arms with the different angles (90°, 80°, 70°, 60° and 50°) are presented in (c). The unpicked fruit are colored in black.
List of Tables

Table 4.1  Optimization variables (bold) and constrained parameters defining the robot kinematics for the considered robot structures. 42
Table 4.2  Performance cost function for the optimal robots with different structures and different numbers of locations (Nloc) around the tree for the Tall Spindle apple tree model. 49
Table 4.3  Performance cost function for the optimal robots with different structures and different numbers of locations (Nloc) around the tree for the Y-trellis apple tree. 50
Table 4.4  Performance cost function for the optimal robots with different structures and different numbers of locations (Nloc) around the tree for the Central Leader apple tree. 51
Table 4.5  Comparison of optimized robots kinematics and kinematics taken from industrial robots. 55
Table 5.1  Fragment of the fruit–location map for the TS apple tree model and RRP robot. 64
Table 5.2  Fragment of the fruit–location map. 65
Table 5.3  Fragment of the fruit–location map with location combinations. 65
Table 5.4  Fragment of the fruit–location map with combination of two locations (Nloc = 2). The numbers of the locations in the combinations are given in the far-right columns. 66
Table 5.5  Fragment of the fruit–location map for the Tall Spindle apple tree and RRP robot including the costs for fruit picking from locations 7 and 10. 69
Table 5.6  Fragment of the fruit–location map with two locations, where the cost-function values for location 19 are lower than those for location 18 for most of the fruit. 70
Table 5.7  Comparison of the number of location combinations for the complete search, search after division into sectors, search with the branch and bound (BB) algorithm without bounding tolerance, and search with the BB algorithm with bounding tolerance \( t_b = 10\% \). 71
Table 5.8  Comparison of the fruit–location maps for Algorithm 1 and Algorithm 2. 74
Table 6.1  Comparison of the optimal robots based on the actual environment (RAE) and on a characteristic environment (RCE) with extreme characteristic targets. 88
Table 6.2  Comparison of the optimal robots based on the actual environment (RAE) and on a characteristic environment (RCE) with mean and extreme characteristic targets. 90
Table 6.3  Comparison of the optimal robots based on the actual environment and those based on a characteristic environment with different numbers of clusters by the proximity indices (shaded cells with \( dF > 10\% \)). 103
Table 6.4  Relative difference, \( dF \), between the performance of the optimal robots based on the actual environment and characteristic environment (RAE and RCE, respectively), and RAE and a robot based on the environment 104
consisting of the worst cases (RWCE) in an actual environment for the Central Leader apple tree model.

Table 8.1 Comparison of optimal robots based on actual and characteristic environments (RAE and RCE, respectively) by proximity indices for the three tree models.

Table 8.2 Performance and optimization time for optimal robots based on a characteristic environment for complex environments.

Table 9.1 Robot performance cost function $F$ for optimal robots for different environments.

Table D.1 Performance of the experimental PPP robotic arm depending on $\theta_2$. 

130

131

135

179
Abstract

Robotic manipulators can perform a variety of agricultural tasks. However, despite decades of research, few agricultural robots have been commercialized. One of the reasons for the lack of agricultural robots on the market today is their high cost, which makes them unprofitable for farmers.

To decrease the cost of robot manipulators for agriculture by improving the robot performance, we propose designing a robot that is optimal for a specific task. In the optimization process, the robot's performance is maximized keeping the ability of the robot to perform the task. To achieve a reliable result, the actual field task must be described and modeled with sufficient precision. However, the complex and unstructured environment of agricultural tasks complicates the task description as well as the robot-design process.

The main goal of this research is to develop methods for simultaneous design of orchard and harvesting robot. The orchard is the geometrical environment of the agricultural tasks, and its influence and interaction with the task-based robot optimal design was analyzed. This analysis allows to simplify the task description by characterizing the environment during the simultaneous design of the robot and its environment. To achieve this goal, the research was conducted to achieve the following specific goals:

- building geometric models of actual agricultural environments
- performing the robot task-based optimization
- developing a methodology for analysis of the agricultural environment, which includes characterization of the environment with the help of our newly formalized concepts of fruit clustering and reaching cones
- testing the fit of the agricultural environment to robotic operation and designing an improved agricultural environment simultaneously with the robot design.

The main results of the research are as follows. For the task-based robot optimization, we created a library with approximately 20 plant models. Software for evaluating the robot's performance effectiveness (optimization of cost function) was written and used for the optimal robot design.
Based on the model library and the software, we found robots with optimal kinematics for a number of agricultural tasks and environments. During the robot optimization, we found that the level of the complexity of environment does not permit our software to solve the optimization problem in an acceptable time. In addition, a method for optimal robot location was developed.

To solve the robot-optimization problem for complex environments with the help of our or any other software, we developed a method for characterizing the agricultural environment by fruit clustering and reaching cones. The method systematically reduces the complexity of the environments, thereby decreasing the amount of calculations and providing a near-optimal solution. The method was approved and successfully applied to complex environments, solving the optimization problem in hours, rather than after days or weeks of calculations. The expected precision of the achieved solutions was 10% in our case.

We made a preliminary design of the robot working environment. We found an environment that was maximally fitted to the robotic operation and optimized one of the parameters defining the structure of the environment.

In general, we developed a set of tools and methodology for analysis and design of the agricultural environment together with the robot design. This methodology is novel in robot design, particularly in agriculture. It improves the robot performance while producing low-cost robots affordable for farmers. The methods developed in this research are applied on apple harvesting, though, they can be used for robotic harvesting of any type of fruits, for other agricultural tasks or any robotic area, where the robot-environment design is used.
List of Symbols and Acronyms

$\alpha_{allow}$  percentage of fruit allowed to be unpicked

$\alpha_{uf}$  percentage of unpicked fruit

AE  actual environment

BB  branch and bound

CE  characteristic environment

CL  tree trained by the Central Leader training method

$D_{clust}$  maximal size of a cluster

DOF  degrees of freedom of robotic arm

$E$  energy consumed by the robot’s actuators

dF  relative difference between robot performance functions

DH  Denavit–Hartenberg

$dl$  relative difference between robot kinematics

$F$  robot performance function, optimization cost function

GA  genetic algorithm

$N_{allow}$  number of fruit allowed to be unpicked

$N_{clust}$  number of clusters constituting an AE

$N_{DOF}$  number of degrees of freedom of robotic arm

$N_{far}$  number of farthest extreme characteristic fruit

$N_{fruit}$  number of fruit on a plant
$N_{loc}$ number of robot locations around a tree

$N_{mean}$ number of mean characteristic fruit equal to the number of clusters

$N_{near}$ number of nearest extreme characteristic fruit

RAE optimal robot based on the actual environment

RCE optimal robot based on the characteristic environment

$q$ vector of angles in the robot joints representing a specific robot configuration

$\ddot{q}$ vector of accelerations of the angles in the robot joints

$t_b$ bounding tolerance in the robot location search

TR time ratio between the computation runtime for solving the optimization problem for the RAE and the runtime for solving the optimization problem for the RCE

TS tree trained by the Tall Spindle training method

YT tree trained by the Y-trellis training method

$W$ power of the robot’s actuators
1. Introduction

1.1. Problem description

Automation in agriculture should improve the farm's profitability and solve the problem of a growing need for agricultural production. A variety of agricultural tasks, which do not require precise treatment, have been successfully automatized (by combines, sorting machines, etc.). However, precise agricultural tasks requiring “personal” treatment and decision-making, such as fruit picking or selective branch pruning, still have no commercial solution (Bac et al., 2014), despite efforts made to create commercial agricultural robots by both academia and private developers. One reason for the lack of success in creating commercial agriculture robots is insufficient cost effectiveness (Bac et al., 2014).

The profitability of farms, for example, an apple-growing farm, can be improved by introducing agricultural machinery such as robotic arms, thereby increasing the cost effectiveness of the growing process (Bechar and Vigneault, 2016). This cost effectiveness depends on multiple factors. Among them are the performance of the agricultural robot and the suitability of the agricultural environment for the robot work, which are the main topics of this research. The effectiveness of the agricultural robot and its environment is improved by their optimal design.

Optimal design of a robot for agricultural tasks is a complicated problem. A cost function representing such problems typically has a large number of optimization variables. A number of sub-problems must be solved during the cost-function evaluation, such as inverse kinematics, robot trajectory planning and optimal locating of the robot. The environment of the agricultural tasks is complex and unstructured, consisting of a large number of objects (Bechar et al., 2010). All of this makes the calculation of the optimization cost function a time-consuming operation. If the number of objects is too large, which is typical for agricultural environment, combinatorial explosion occurs during the solution of the optimization problem, and the solution time becomes intractable.

Geometric features of the robot’s environment strongly influence the robot performance (Bac et al., 2014). The agricultural robot's environment consists mainly of plants, with geometry defined by biological rules and agronomic conditions. In addition, the agricultural environment is
dynamic (growing and being shaped), unordered (without repeating patterns) and irregular (without patterns), calling for the description of a large number of cases. Thus, design of the robot’s environment is an algorithmically and computationally complex problem.

To simplify the design of agricultural robots, the actual robot’s environment is often markedly simplified, resulting in either a schematic description without details or a description that only takes the worst case into consideration (Han et al., 2007, Van Henten et al., 2009). In the former, where the robot design is based on such a simplified environment, the suitability of the resultant robot cannot be guaranteed, whereas in the latter case, the design yields a universal robot suited to a wide range of tasks at the expense of its simplicity, leading to reduced cost effectiveness.

1.2. Objectives

The main objective of this research is developing a methodology to perform the optimal design of a robot and its working environment. This objective has four specific objectives: geometric description of the agricultural environment, showing the significance of agricultural robot optimization, decreasing the time for optimal robot design, and optimizing the working environment simultaneously with the robot optimal design.

The result of the optimal robot design depends on the description of the robot working task and environment. The precision of the environmental modeling defines the amount of information used in the optimization.

The first objective of the research is to achieve a sufficiently detailed description of the actual agricultural environments.

The second objective of the research is to show the importance of the methodological search for a robot that is optimally fitted to perform its task under realistic conditions, and of using a detailed environment model for this search.

Robot optimization is a complex problem, especially for unstructured agricultural environments. The third objective of the research is to develop a methodology for environment analysis and characterization decreasing the solution time of the robot optimization problem while retaining solution precision.
The working environment of the robot influences the robot's performance. The **fourth objective** of the research is testing the fit of the agricultural environment to robotic operation and designing an optimal agricultural environment simultaneously with the robot design.

### 1.3. Significance

One of the reasons for the lack of commercial agricultural robots is their high cost (Bac et al., 2014), making them unobtainable for farmers. Their cost effectiveness can be increased by effective design of the robot and its environment. This research provides tools and a methodology to design a robot with optimal kinematics and an optimal environmental structure. This will help to improve the performance of the robotic system, which will advance agricultural robots toward commercialization and their application in farms.

In this research, we built a library of models of plants growing in orchards as well as geometrical simulations of the plant structures. The models of actual and simulated plants were used as an environment for the optimized robots to achieve the objectives of this research. This library and the methodology for plant modeling and simulation can be used in further research in agricultural robotics.

We show the differences in robot performance when robot design optimization is based on i) a detailed model of the environment or ii) a simplified model of the environment, or when iii) robots are not optimized at all. We show that to achieve a near-optimal robot, the environment simplification should be performed according to the proposed guidelines.

We developed a new methodology for characterization of the robot's working environment that helps decrease the solution time for problems involving a large number of objects. We used this method for robot optimization based on a complex agricultural environment. This method can be used in any research involving complex environments.

We show the interaction between robot and its working environment. We propose methods to improve the robot performance by testing existing environments and designing an optimal one simultaneously with the robot optimal design.
The methods developed in this research are applied on the apple harvesting, though, they can be used also for robotic harvesting of any type of fruits, other agricultural tasks or any robotic area, where the robot-environment design is used.

1.4. Innovation

This research presents a number of innovations in the design of a robot for agricultural tasks, optimization of robot kinematics, and optimization of the robot's working environment.

- Building a library of plant models and comparing their geometrical structures are novel approaches for robot design and botany.
- Robot optimization based on an exact tree model is new in agricultural robot design.
- Methodological simplification and characterization of the robot’s working environment is new in robot design.
- The multiple-robot-location method is new in robot design.
- The reaching-cones method and the analysis of the robot's reachability map are new in robot design.
- The analysis of existing environments and the design of an optimal agricultural environment with respect to robot performance are new in agricultural robot design.

1.5. Contribution

The research makes a number of contributions to the design of robot kinematics and working environments.

- The optimal robot design can help to improve the robot performance during design of low-cost robots that are accessible for farmers.
- The library of the detailed plant models can be used for research in agricultural robotics.
- The methodology for environment characterization can help achieve accurate solutions when the robot-design problem is complicated and must be simplified.
- Analysis of the agricultural environment can help find or build an environment, thereby improving the robot's performance.
1.6. Thesis structure

Chapter 2 provides the literature review. Chapter 3 presents the main scope, methods and assumptions of the research. Chapter 4 describes the task-based robot optimization for agricultural tasks and demonstrates its significance. Chapter 5 describes the optimal robot location problem, which is part of the task-based robot optimization problem, and develops a method for its solution. The methodology for characterization and simplification of the agricultural environment is developed in Chapter 6. This methodology is generalized and applied in Chapter 6.7 to optimization of a robot for actual environments measured in orchards. The idea of a characteristic environment is generalized for robot obstacles in Chapter 7 by developing reaching cones, representing obstacle-free areas where a robot can move without collisions. In Chapter 6.7, we apply all of the described methods to characterize an environment consisting of models of single trees and tree rows, and perform robot optimization based on the simplified environments. The idea of designing an agricultural environment is considered in Chapter 9. The existing and simulated agricultural environments were analyzed for robotic operation. Appendix A describes the plant model library, consisting of models of trees and plants. Appendix B contains calculations for the environment-characterization analysis. Appendix C describes an analysis of the geometrical similarity of same and different environments. A field experiment was conducted to prove the concept of the task-based design, and is described in Appendix D. The relation between the thesis chapters is presented in the flowchart in Fig. 1.1.

3. Introduction and assumptions
4. Robot kinematics optimization
5. Robot location optimization
6. Environment characterization
7. Reaching cones
8. Robot kinematics optimization based on environment characterization
9. Environment design

Fig. 1.1 Relation between the thesis chapters.
2. Literature Review

2.1. Robot design

2.1.1. Optimal robot design

Optimal robot design is common in the industrial domain and has been considered in a number of studies. Wunderlich (1991) designed a robot covering the working volume inside a box modeling the interior of a vehicle. Paredis and Khosla (1996) considered a task in a highly restricted space. Leger and Bares (1999) used exact task environment and motions for satellite maintenance. Kim and Khosla (1993) applied the robot design for inspecting the surface of a space shuttle.

The following robot parameters are typically used as optimization variables: number of degrees of freedom (DOF), robot configuration, link lengths (Han et al., 2007, Zhou et al., 2012), actuator type and power (Wunderlich, 1991). The following considerations are used for cost function and constraint definition: covering a given working volume (Bergamaschi et al., 2008, Ceccarelli and Lanni, 2004, Peterson, 1999), avoidance of a singularity region in the robot workspace (Ceccarelli et al. 2005, Stocco et al. 1998), Jacobian conditional number, manipulability and dexterity indexes (Gosselin, 1992).

Most studies offer a description of the workspace and the robot task. Only a small number of studies deal with optimization, taking into account models of the robot workspace (Edan et al., 2000, Han et al. 2007, Paredis and Khosla, 1996). However, several specialized robots have been designed with a consideration of motion trajectory (Van Henten et al., 2009, Zhou et al., 2012).

2.1.2. Optimization runtime

The simplicity of the optimization constraint is important for the optimization runtime, which is one of the main constraints in the robot-design process. The runtime is considered in many studies (Bergamaschi et al. 2008). Typically, the number of optimization parameters is large and, taking into account the complexity of the cost function, this leads to a long optimization runtime. The runtime can be decreased by choosing the appropriate optimization algorithm. Bergamaschi et al. (2008) and Shiakolas et al. (2002) analyzed different algorithms and indicated the most
effective ones for robot optimization. Rubrecht et al. (2011) used a genetic algorithm (GA)-based optimization algorithm. Mann et al. (2015) develop a greedy algorithm that solves the multiple manipulators motion planning in polynomial time. In addition, parallel computing can be used. For example, the complex Darwin2K algorithm developed by Leger and Bares (1999), covering a large number of robot design aspects, was run simultaneously on 30 computers to accomplish the calculation in an acceptable time. To the best of our knowledge, environment characterization as a method of decreasing optimization time has never been addressed. This is one of the goals of this research.

2.1.3. Optimization of the robot environment

Optimization of the robot environment has been well studied, ever since robots began to be employed in industrial manufacturing where it is defined as design of the robotic cell. The main methods used in cell design are effective scheduling, use of multiple grippers, and parallel working robots (Dawande et al., 2007), decreasing the work cell volume (Lounell et al., 2009), and optimization of robot location (Kamrani et al., 2009). A number of techniques for increasing the robot ergonomics were described by Nof (1999). Design of the industrial environment can be complicated and challenging. Nevertheless, it typically involves only a small number of objects and DOF. In addition, the industrial environment is usually not restricted in its shape or configuration. Thus, optimization of the industrial environment is simpler than designing environments involving humans and other biological entities. For example, design of a house for human–robot symbiosis (Sugano et al., 2006) is a complex state-of-the-art project.

2.2. Robot design in agriculture

Existing agricultural robots have been reviewed by Bac et al. (2014) and Bechar (2010). These reviews characterize the crop environment that is relevant for robotic harvesting and formulate challenges and directions for future research and development. One of the challenges in agricultural robotics is simplification of the robot's task.

2.2.1. Task-based robot design in agriculture

The main applications of robots in agriculture include ground vegetable picking, e.g., harvesting of radicchio (Foglia and Reina, 2006), mushroom (Reed et al., 2001), melon (Edan, 1994, Mann et al., 2015), lettuce (Cho et al., 2002), and watermelon (Sakai et al., 2008, Umeda et al., 1999);
greenhouse vegetable harvesting, e.g., cucumber (Van Henten et al., 2009), tomato (Monta et al., 1998), strawberry (Feng et al., 2008, Agrobot (agrobot.com)), and eggplant (Han et al., 2007); chrysanthemum cutting (Kondo and Monta, 1995), treatment of greenhouse plants (Belforte et al., 2006, Kondo and Ting, 1998) and fruit harvesting, e.g., apple (Baeten et al., 2007, Peterson et al. 1999), orange (Edan et al., 2000, Flood et al., 2006, Lee and Rosa, 2006), cherry (Tanigakia et al. 2008) and kiwi (Flemmer, 2009). The agricultural robots are usually designed for one specific task. The designers take into account the distinctive features of each task and fit the robot to them, achieving relatively simple and low-cost robots.

Task-based robot design usually relies on general guidelines, without optimization. Sakai et al. (2005) designed a robot with a 2.8-m reach that could hold a 15-kg payload and was used for watermelon harvesting. Belforte et al. (2006) constructed a robotic arm for greenhouse applications “with standard components and simple home-built parts”. Scarfe et al. (2009) used a “light, simple and cheap” arm for kiwi harvesting. Cho et al. (2002) used pneumatic driving in a robot for lettuce harvesting. De-An et al. (2011) described requirements for an apple-picking robot “sufficient to perform the harvest operation”. All of these robots were lightweight, had a low number of DOF and were actuated by simple actuators: all features that simplified their structure. However, the kinematics of these robots was not systematically fitted to the features of their tasks, and therefore, their optimality could not be guaranteed.

2.2.2. Optimal robot design in agriculture

A precise approach to task-based design is to conduct task-based optimization of the agricultural robots. Edan et al. (1994) evaluated the performance of an agricultural robot for different types of robots, number of arms, multiple arm configurations, workspace design and dynamic characteristics. Han et al. (2007) designed an optimal robot for eggplant harvesting. The environment was defined as the “required rectangular working space” based on the “common plant height, fruit distribution, and circumference”. The robot’s workspace containing this environment was minimized. Van Henten et al. (2009) performed robot optimization for cucumber harvesting. The task construction was based on the “most difficult cases”. Although these tasks, which are taken as the constraint for optimization, are based on the features of the actual agricultural tasks, they are not sufficiently effective for robot optimization. A “worst-case”
approach leads to a “universal” rather than optimized robot. In addition, both environment modeling and a statistical analysis are required to design a robot that is suited to its actual task environment. In this research we perform the robot optimization based on models of actual environment.

2.2.3. Robotic apple harvesting

Apples are among the most popular crop fruits for the development of robotic harvesters (together with oranges, sweet peppers and tomatoes). Complete single-armed robots for apple harvesting were developed by De-An et al. (2011) and Baeten et al. (2008). A human-operated robotic arm with kinematics fitted to the task was developed by Luo et al. (2015). Parts of the robotic system have been developed in numerous studies, such as a detaching mechanism by Bulanon et al. (2010), fruit-detection systems by Fernandez et al. (2014) and Linker and Kelman (2015), and a motion-planning algorithm for apple harvesting by Nguyen et al. (2014). However, the robotic harvesters are still too complex and their cost is still too high for commercialization.

End effectors for fruit picking

Numerous end effectors have been designed for the harvesting of different fruits by different methods. Van Henten et al. (2002) constructed a thermal cutting device for cucumbers, Bulanon (2010) developed a rotating wrist to detach the apple fruit by breaking the peduncle, and Kondo et al. (2010) developed an end effector for harvesting tomato fruit clusters. A methodology for the end effector design based on the graspability mas was developed in the CROPS project (Eizicovits and Berman, 2014). The variety of developed end effectors allowed us to assume that there exists an end effector suitable for the task considered in this research.

2.3. Tree modeling and design

2.3.1. Tree modeling

Environment modeling is a critical step in environment characterization. Several studies have conducted environment analyses using accurate agricultural environment models. Edan et al. (1991) measured the location of orange fruit to achieve an efficient trajectory for a specific robot. Lee and Rosa (2006) measured the location of orange fruit to develop a fruit-picking technique.

There are three commonly used methods for plant reconstruction and modeling:
1) based on visual images (Santos et al., 2013; Berge曼 et al., 2012)
2) 3D scanning (Preuksakarn et al., 2010; Emery et al., 2010; Méndez et al., 2014; Cheein et al., 2015)
3) mechanical sampling of points in space (passive robot device used by Edan et al., 1991).

The image-processing methods (1 and 2) are relatively rapid, but they cannot provide exact modeling because of the limitations of image processing. The mechanical sampling method provides a detailed and exact model, but the modeling process is time-consuming.

2.3.2. Plant simulation

To obtain a sufficient number of tree models for successful robot optimization, these models can be simulated, instead of the exact modeling described earlier. Simulation of plant geometrical structure is mainly based on the L-Systems method (Prusinkiewicz and Lindenmayer, 1990) for the geometrical features and on a Markov-chain method (Costes et al., 2006) for the statistical behavior of the trees. These methods are used in plant-simulation software, such as L-studio (Karwowski and Prusinkiewicz, 2004), PlantToon (Bonora et al., 2013), MAppleT (Costes et al., 2008), and AMAPstudio (Griffon and Coligny, 2014). All of these simulation tools allow the construction of realistic models of trees, including such aspects as tree growth, branch pruning and gravitational influence. These simulation tools can be used for robot optimization after adapting them to the robot-optimization needs and collecting data for the tree modeling.

2.3.3. Plant-training systems

Modern high-density plant-training systems, such as the Tall Spindle and Y-trellis, were developed mainly to increase fruit yield and quality (Robinson et al., 1991) using advanced agro-techniques. In addition, they conserve labor time during harvesting by providing a convenient environment for human harvesters. This advantage can also provide an environment that is suitable for robotic harvesters, turning them into a profitable harvesting solution. The goal of this research was to evaluate the fit of these training systems to robotic harvesting.

2.3.4. Optimization of the agricultural environment

Simplification and structuring of the agricultural environment have been investigated by a number of researchers. Hua et al. (2003) used an L-systems tree simulation to find the optimal
parameters for maximal light absorption by the tree canopy. Taugourdeau et al. (2012) analyzed the optimality of the morphogenetic features of trees. Edan et al. (2000) tested the effectiveness of the robot motion trajectories for different type of trees. Nevertheless, environment design or optimization for robotic harvesting have never been performed. Such optimization is difficult because of the large number of optimization parameters. The difficulty lies in reaching the desired design, the extensive work required to design a tree, and the required knowledge of the plant's behavior.
3. Methodology

In this research, we made use of a number of methodologies: field data collection and modeling of plants, simulation of robot performance, optimization of robot and environment structures, statistical analysis of the geometrical features of the environment, and field experiments to validate the method.

3.1. Field data collection and modeling of plants

The goal of the research was to develop a methodology to design an optimal robot for agricultural tasks. Therefore, modeling of the task environment is crucial. To measure and model the robot’s environment, we developed a mechanical measuring device, a 'digitizer'. It is to be specific Fig. 3.1, and the prototype is shown in Fig. 3.2. This digitizer is a passive robotic arm with a kinematic structure of an RRRR robot (four revolute joints) with three links at lengths of 1.2 m, 1.2 m, and 0.4 m. The angles of the structural joints are measured by encoders with a resolution of 0.07°, resulting in a total maximum error of 3.5 mm. The location of the tip of the digitizer is calculated according to the forward kinematics of the robot. A full measurement of a cultivated apple tree (bounded by a box of 2x2x3 m) using this device takes approximately 3 hours.

![Fig. 3.1 Drawing of the measuring devise with dimensions in mm and a list of the parts.](image)

<table>
<thead>
<tr>
<th>Item No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Tripod</td>
</tr>
<tr>
<td>2</td>
<td>First Axis</td>
</tr>
<tr>
<td>3</td>
<td>Second Axis</td>
</tr>
<tr>
<td>4</td>
<td>First Link</td>
</tr>
<tr>
<td>5</td>
<td>Third Axis</td>
</tr>
<tr>
<td>6</td>
<td>Second Link</td>
</tr>
<tr>
<td>7</td>
<td>Fourth Axis</td>
</tr>
<tr>
<td>8</td>
<td>Third Link</td>
</tr>
<tr>
<td>9</td>
<td>Tip</td>
</tr>
</tbody>
</table>
Fig. 3.2 *The digitizer, a mechanical measuring device, is a passive robotic arm with RRRR structure.*

The coordinate system of the plant environment is defined as follows: $Y$ axis is the tree row line, $Z$ axis is upwards, perpendicular to the ground, and $X$ axis complements the right-hand system (Fig. 3.3 a, b). The origin of the coordinate system is at the point where the plant trunk intersects the ground.

Fruit location is defined as the point in the center of the fruit. We assumed that the fruit's central point is located in the middle of the fruit orientation vector, starting from the point at which the fruit stem is connected to the holding branch, toward the opposite pole of the fruit (Fig. 3.3 c).
Fig. 3.3 Views of agricultural environments: (a) row view and (b) side view of an apple tree shaped by the Central Leader method. The coordinate system is shown by arrows. The location of the fruit is defined as shown in (c).

The environment of a robot in agricultural tasks consists mainly of plants, e.g., trees in an orchard, or plants in a greenhouse or open field. In this research, the following features were modeled: branch length, location, orientation and thickness; fruit location, orientation and size; trellis wires, and the ground plane. In the model, tree branches were divided into straight intervals and represented by cylinders. The fruit were represented by oriented ellipsoids. The ground was modeled as a horizontal plane.

3.2. Simulation of robot performance

A robotic arm is a complex mechanism. Its performance can be described by its geometry, kinematics, and dynamics. In this research, the robot kinematics was defined by the Denavit–Hartenberg (DH) method. To simulate the robot's collision with obstacles, the geometry of the robot was described with the help of geometrical primitives: the robot's links are represented as cylinders. The dynamics of the robot was simplified: the robot joints move with constant accelerations and decelerations defined by the power of the robot actuators, and the mass and moment of inertia of the robot links. The power of the actuators is predefined and constant. The mass of the link is proportional to its length.
All calculations were programmed and performed in MATLAB™. These calculations included: optimization, geometrical calculations, simulation of robot motion and collision, and calculation of the torques in the actuators and energy consumption.

3.2.1. Inverse kinematics

In this research, we considered robotic arms with three DOF. Industrial robots with this number of DOF are designed such that their inverse kinematics can be described analytically. This is achieved by setting some kinematic parameters equal to zero, to simplify the solution of the inverse kinematics problem. To consider all of the possibilities of the robot kinematics, we included all possible discretized values of the kinematic parameters taken from predefined intervals. This makes the expressions of the inverse kinematics too complicated to be solved analytically. Therefore, the inverse kinematics problem for all types of robot structures was solved numerically.

The equation system of the inverse kinematics problem was solved using Newton’s method for non-linear equation systems. The expressions for the equation system Jacobian were calculated separately for all considered robot structures. The termination criterion was set to an accuracy of 0.001 m in the robot’s end-effector location in the workspace. A trial and error method derives that to achieve all possible solutions, a uniform grid that includes four initial guesses for each variable (a total of $4^3 = 64$ initial guesses) is sufficient.

3.2.2. Trajectory planning

The trajectories between the robot's home position and its targets (fruit) are found with the help of trajectory planners. The complexity of the environment does not permit using a deterministic planner such as A* or Potential Fields. Thus, a probabilistic rapidly exploring random tree (RRT) algorithm (LaValle, 2006) was used. Our implementation of the original RRT used 100 vertices and an incremental distance of 0.03 m.

3.2.3. Robot collision

To check the collision between the robot parts and the environment, geometrical models of the robot and the environment were used. The robot links are presented as cylinders with the
following diameters: 0.2 m for the first link, 0.15 m for the second, and 0.1 m for the third. The environment modeling is described in Appendix A.

3.3. Optimization of robot and environment structures
The optimization was the most time-consuming computational operation in this research and in robot task-based optimization design in general. One of the main goals of the research was to decrease the runtime of the optimization. Thus, choosing the appropriate optimization method was crucial: grid search, genetic algorithm (GA) and simplex methods were compared by a number of criteria, and the grid search method was found to best suit this research.

3.4. Statistical analysis of the geometrical features of the environment
A basic statistical analysis was used for characterization and comparison of environment. The k-means clustering algorithm was used for the partition of environment objects into groups. The Kolmogorov–Smirnov test was used to compare the distributions during the tree similarity analysis.

3.5. Field experiment
We used a robotic arm for apple harvesting provided by FFRobotics Ltd. (www.ffrobotics.com, Israel), with the optimal structure for performing a proof-of-concept harvesting task. The optimality of the designed arm was verified in a field experiment, showing the effectiveness of the task-based robot design.

3.6. Approach and assumptions
To achieve the objectives of this research, we performed the following steps. To develop a method for the environment characterization, we analyzed the structure of the agricultural environment, developed a new geometrical feature of the environment (reaching cones), and formulated the guidelines for its characterization. To design the agricultural environment, we used existing and simulated environment models and tested their fit to the robotic harvesting. To find a robot with optimal kinematics, we defined the performance cost function and constraint of the robot-optimization problem, defined the robot's agricultural task, and developed a method for determining the optimal multiple robot locations.
3.6.1. Cost function for robot optimization

The cost function for the optimization of an agricultural robot is complex and consists of a large number of factors. A single universal parameter for the robot optimization can be profitability for the agricultural farm using the robots, e.g., the difference between the investment and the farm's income. However, this study could not deal with all of the economic factors constituting the investment and the income, and therefore the cost function had to be simplified to the level of the kinematics of the robot and geometry of its environment, which are the topics of this research. To simplify the optimization of the cost function, we made a number of assumptions.

Assumption about the factors not taken into account

The aspects of the robot performance related to the computer vision and image processing were developed in numerous other works and were beyond the scope of this research.

The influence of the economic factors on farm profitability differs for different countries, economic situations, farm types, etc. Among the main economic factors are cost of agricultural production, cost of labor working time, cost of performance of the agronomic tasks, cost of electricity, cost of robot manufacturing and maintenance. We did not use these factors in our research, although we did transform the economic factors related to the robot's cost into factors defined in robotics.

Assumption about robot profitability

The overall profitability of a farm using agricultural robots consists of agricultural and robotics factors (described in the previous paragraph). We assumed that the agricultural and robotics factors have unconnected effects on the overall profitability, and that the robotics factors can therefore be considered separately.

Assumption about the robot's initial and maintenance costs

The cost of a robot consists of its initial cost and maintenance cost. We assumed that the main mechanical components affecting the initial cost are the cost of manufacturing the robot, depending on the required accuracy, and the cost of the actuators, depending on their number,
type and power. In addition, the main mechanical components affecting the maintenance cost were assumed to be **energy consumed by the robot** and **performance time**.

**Assumption about the actuator’s power**

To simplify the cost function, we predefined the power of the actuators. This decreases the number of optimization variables, and allows us to normalize and compare the efficiencies of robots with different kinematics.

**Assumption about robot manufacturing**

It is difficult to evaluate with high accuracy the initial cost of a robot, though, we assume that there is a dependence between the initial cost and performance time. For a robot with a specific kinematics, a single robot performs a task in a specific time, whereas a number of identical robots might perform the task in a shorter time. To study this dependence, we solved the optimization problem for a number of robot locations, which is equivalent to the optimization problem with a number of robots assuming that the robots do not cooperate.

**Assumption about energy consumption**

We assumed that the cost of consumed energy is negligible relative to the cost of performance time. We did not include energy consumption in the robot maintenance cost, although we did use it to calculate the performance time when the power of the robot actuators was predefined described in Chapter 4.

**Assumption about robot performance time**

Referring to the industrial domain, the performance time, or time needed to perform a working cycle, is the most critical aspect of the robot's cost effectiveness. Similarly, the performance time is critical in agricultural tasks, especially in seasonal tasks such as fruit harvesting, which must be completed in a specific time.

Using all of these assumptions, we defined the optimization cost function describing the robot's cost as the robot's performance time. The mathematical definition of the optimization cost is given in Section 4.1.
3.6.2. Constraint for robot optimization

The environment of the agricultural tasks is complex and unstructured, causing two principal limitations in the robot's performance: difficult object recognition and difficult object reaching. These limitations dictate the robot's minimal profitable rate of success (Bac et al., 2014), which is the percentage of successfully treated robot targets. For example, in human fruit picking, farmers define some percentage of fruit that can remain unpicked because of their problematic location on trees. The same percentage of unpicked fruit can be defined for robotic harvesting. We defined the optimization constraint as the condition for which a robot is able to pick a number of fruit larger than some predefined number.

3.6.3. Performance in different environments

Similar to any mechanism, the robotic arm is designed to perform a set of tasks. In the domain of industrial robots, every company manufacturing robots classifies them according to their task: general, palletizing, welding, painting, etc. The robots in each of these classes have different parameters and in particular, different kinematics.

Assumption about agricultural environment

In agriculture, similar tasks, such as fruit picking, can differ in different environments. For example, orange, pear and mango trees (Fig. 3.4) have different shapes and dimensions, defining different harvesting tasks and consequently different requirements for the robot structure. Most orange fruits grow on the external layer of the tree. Thus, the robot can move its end effector to the fruit location along obstacle-free trajectories. In contrast, pears grow along the branches. Thus, to reach them, the robot has to penetrate the tree canopy and avoid collisions with branches. Finally, mango fruits grow in the lower part of the tree with a near vertical orientation. Thus, unlike the previous cases, the robot can be relatively small and its end-effector orientation can be constant. Thus, different environments may lead to different optimal robots.

Considering the complexity and variability of the agricultural environments, we assumed that the robot is designed to perform a task in a single orchard with trees of the same type trained by the same training method. The tree training includes pruning of branches and tying the branches to trellises.
3.6.4. Environment characterization

In the fruit-harvesting task, the robot environment consists of trees with branches and fruit, trellises and the ground. The simplest apple tree modeled in this research had 30 fruit and 30 branches. More complicated tree models have hundreds of fruit and up to 1000 branches. Solution of the inverse kinematics and motion planning of the robot in such a complex environment takes significant time, making the optimization process difficult and inefficient.

Assumption about the characterization of orchards

Ideally, to design an optimal robot for fruit picking in some orchard, all of the trees in the orchard must be modeled, describing all possibilities for fruit approach and collision avoidance. However, since such an exact description demands too much time and effort, we assumed that the orchard could be characterized by a number of individual trees, similar to statistical sampling of a population. Thus, to find a robot that is optimal for an orchard, we find a robot that is optimal for a group of trees characterizing the orchard. The correctness of this assumption and the required number of characterized trees are discussed in Appendix C, where different environments are analyzed and compared.
Guidelines for environment characterization

Notably, if the sampling data are too large, a combinatorial explosion is caused while evaluating the robot performance cost function. Therefore, only a simple environment model including a small amount of data (in this research up to 100 geometric objects) characterizing the actual environment (AE) is effective for robot optimization. This simple artificial environment is defined in this research as the characteristic environment (CE). The amount of required data is discussed in Appendix C.

Construction of the CE is based on the features of the AE. On the one hand, the CE must include the required information affecting the optimization solution. On the other, it must be the simplest environment that still provides a sufficiently close description of the AE to decrease the optimization runtime. The main intuitive guidelines for defining such a CE are as follows:

1. Use typical cases, avoiding worse or atypical cases that tend to lead to the design of a universal robot, not an optimal one.
2. Use the minimal amount of necessary information when defining the environment.

Environment characterization helps simplify the description of the environment: the number of objects making up the environment decreases tens of times. This makes the robot optimization process up to ten times faster and possible in complex cases.

3.6.5. Optimal locating of robots

Placing a robot in optimal locations is part of the optimal robot design. Usually, in industrial environments, the robot is placed in a single location, for example, stationary robots working on conveyor tasks. Nevertheless, if robot targets are distributed over a large volume, such as the volume defined by a tree, the robot cannot reach them from a single location. The robot has to move on a platform vehicle between a number of locations, each optimal for approaching a specific set of targets. This case is typical for agricultural tasks, where a robot has to reach targets throughout the area of an orchard or greenhouse. The influence of the motion of the robot platform on the total robot cost function must be evaluated with respect to a given platform, which complicates the optimization problem. To simplify the problem, we used an assumption that renders the problem independent of the platform.
Assumptions about the robot platform's motion time

We assumed that the robot cannot move its arm and perform picking while the platform is in motion. The stopping and starting of the motion can take a certain interval of time, termed stabilization time. The following processes can be included in this time: stabilization and vibration damping after stopping, creating a target map for the new location and ordering the targets, robot platform acceleration from rest (especially if the load is being carried by the robot platform), etc. Because of the large number of unknown parameters, the stabilization time cannot be calculated in the general case; it must be supplied by the robot designer. We assumed that the stabilization time has the same value for all robot structures and all trees.

3.6.6. Reaching cones

The idea of reaching cones was derived during the analysis of reachability of the fruit for robotic arms. This idea is one of the most significant improvements of the robot design process proposed in this research. If there exists some obstacle-free space around a fruit, the fruit can be reached by the robot moving its end effector (together with the robot link holding it) inside this space. The fruit itself represents a boundary point of this space, assuming that the fruit is located on a branch representing an obstacle that is not included in the space. Knowing the typical sizes of the reaching spaces, we can draw conclusions about the robot kinematics and optimal location for reaching a fruit. Hence, we can create a description of characteristic reaching spaces before the robot optimization and independently of the robot itself. Since the robot specifications are unknown during the robot optimization, the robot navigation problem cannot be solved in the robot configuration space. In this case, the reaching spaces have a clear advantage, since they can be built in the robot workspace, and not in the robot joint space.

In this fruit-reaching space, we include all rays starting from the point of the fruit location and continuing without collision with obstacles. In this research, we propose characterizing the reaching space by circular cones with the fruit at their vertex. The cone is a geometrical object that is suitable for describing the reaching space, having a small number of defining parameters: the cone opening angle, the vector of direction of the cone axis and the cone height.
We used the reaching cones to analyze the robot environment. To solve the robot-location optimization problem, we used fruit reachability, defined as the number of fruit that can be reached by a robot from any point in the robot workspace around the tree. The fruit reachability was calculated using the reaching cones. To solve the robot optimization problem, we used the CE, including the obstacles, which was constructed using the reaching cones.

### 3.6.7. Environment design

One of the useful applications of task-based robot optimization is the design of the robot's working environment. In this research, the environment design consisted of geometrical features of the objects composing the environment. In the example in Section 3.6.3 we saw that the environment defines the kinematics of its optimal robot. Consequently, the cost of the robot depends on the environment.

Thus, we can find not only the optimal robot for its environment, but also the environment that is optimal for a robot. More precisely, the robot and environment are optimized simultaneously, or the cost of the robot–environment system is optimized. This optimization problem is much more complicated than the robot optimization, since it includes the agronomic aspects of the environment design. As a preliminary stage, we created a “discrete” solution for this problem: we modeled a number of existing environments and found the environment for the optimal robot with the minimal cost. Next, we defined a “continuous” problem: we took one of the parameters of a specific environment, changed it continuously (with some resolution) and found the optimal robot for each constructed environment. Assuming that each environment can be constructed with the same agronomic cost, the value of the parameter minimizing the robot cost can be found. If the robot designer has the actual agronomic cost of each environment construction, this assumption can be relaxed and the robot–environment system optimization problem can be solved precisely.

### 3.6.8. Statistical analysis of actual environments

The variability of the agricultural environment is the main cause of the complexity in robot design. As explained above, modeling the entire orchard to achieve the exact robot that is optimal for that orchard is impractical. Hence, we assumed that robot optimization based on a number of trees from that orchard yields a robot which is sufficiently close to the exact optimal
robot. To achieve this, the minimal number of sampled trees that is sufficient for characterizing the entire orchard is evaluated.

To estimate the number of sampled trees, we conducted a comparison of the tree features influencing the optimal robot. These are geometrical features of the environment, representing constraints for the robot optimization, namely fruit location and reaching-cone orientation which characterize the branches and other obstacles. Thus, we could determine the differences in the kinematics of the optimal robots for different environments based on the differences in the feature distributions.

To compare the distributions of the features of the trees from the model library, such as fruit location coordinates, we used the Kolmogorov–Smirnov test. The comparison showed that the difference in features between the trees of different types and shapes is relatively large. However, the difference in features between trees of the same type and tree-training method is also significant.

The exact sufficient number of sampled trees was not found in this research because of an insufficient number of available tree models (maximum five models from different orchards), although this number can be evaluated during the analysis of the environments.

3.6.9. Additional assumptions
In addition to the assumptions made for the formulation of the optimization cost function and constraints, we made the following assumptions to simplify the problem and to focus on the main goal of the research.

1. The robot has a target- and obstacle-recognition system.
2. The robot is carried by a platform that is able to move on a given terrain and stably place itself by a tree or plant.
3. The robot has a motion-planning system creating optimal motion trajectories.

Assumption about the end effector
Since end effectors for fruit picking have been designed and studied in numerous works, we assumed that there exist end effectors for every type of fruit. Since end-effector parts surround a
fruit while picking, they can collide with the branch on which the fruit is growing. To avoid this collision for all types of end effectors, we assumed that the fruit can be picked only from the direction of its lower hemisphere, as shown in Fig. 3.5.

Fig. 3.5 Assumption about the direction of the end-effector approaching a fruit.
4. Task-Based Robot Optimization

The task-based robot optimization for agricultural tasks is an optimization problem with constraints that is used to design a robot with optimal kinematics.

4.1. Robot performance cost function

The effectiveness of the robot is evaluated by the robot performance cost function. The effectiveness depends on the customer’s needs. Thus, the cost function cannot be strictly defined for the general case (as discussed in Section 3.6.1). In this research, we used robot performance time as the cost-function value. We studied the mechanical aspect of the robot's performance, and we therefore considered only the robot motion time, which is defined here as the robot performance cost function $F$. This function depends on the robot's geometry and the power of its actuators.

The performance time depends on parameters such as actuator power and weight, construction material, etc., which are defined by the designer; an exact calculation of the performance time is therefore also nearly impossible to obtain in the general case. To find the optimal robot for our case study, performance time was evaluated by making several assumptions about the robot actuators and structure:

- Robot arm kinematics is predefined as 3 DOF with revolute or prismatic joints and general DH parameters. The number of DOF is denoted as $N_{DOF}$.

- Actuators are considered massless, assuming that the actuators are mounted on the robot base and are transmitting forces through four bar mechanisms or cables, similar to the actuating system presented in Scarfe et al. (2009).

- The length density of the robot links is taken as 1.5 kg/m (similar to the density of a 3-mm thick aluminum tube with a diameter of 60 mm).

- The mass of the load is taken as 0.15 kg (assumed as the mass of an average apple).

- The power of the robot actuators is assumed with specific values: 10 W is the power of the actuator in the first robot joint actuating the weight of the entire robot. The power of the rest of the actuators decreases proportionally to the weight of the links moved by the
actuator. The values are assigned close to realistic values in industrial robot actuator, though, they can be changed according to a considered case.

- Time spent moving the mobile platform from one location to another is not included in the cost function (because of the large variety of platform types). Nevertheless, a designer working with a specific mobile robot can take the moving time into account when evaluating the total robot cost function.

- The robot places the picked fruit in a gathering bin attached to the robot platform. Thus, the robot task must consist of the following stages: moving the end effector from the robot home configuration to a fruit, approaching the lower hemisphere of the fruit, and retracting back to the home configuration. Analysis of the motion required for placing the fruit in the bin is beyond the scope of this research.

- The robot end effector is an apple-picking gripper.

The precise computation of time and energy of motion is cumbersome for mechanical systems with 3 or more DOF. Therefore, we assumed that the time approximation with the help of basic physical expressions would be sufficient in this research. Assuming that the robot geometry, link masses and moments of inertia, and actuator power are known, the time spent picking a specific fruit is

$$t_{fr,i} = \max\left(\frac{E_i}{W_i}\right) \quad \text{Eq. 4.1}$$

where $E_i$ is the energy consumed by the $i$th actuator, and $W_i$ is the power of the $i$th actuator.

The energy $E_{fr} = \sum_i E_i$ needed to pick a single fruit is the energy of the robot’s movement from its initial home configuration, $q_i$, to a final configuration, $q_f$, set for picking the fruit. Therefore, the energy is calculated as

$$E_{fr} = E_d + E_s + E_{damp} \quad \text{Eq. 4.2}$$

where $E_d$ is the dynamic work against the inertia of the robot and the load, $E_s$ is the static work against the load and the robot’s weight, and $E_{damp}$ is the damping work against the friction in the robot joints. The latter depends on the gearbox parameters chosen by the designer, hence it is disregarded thereafter. The static work is evaluated as
\[ E_s = \int_{q_i}^{q_f} \tau(q) \, dq \quad \text{Eq. 4.3} \]

The torques, \( \tau \), produced by the actuators are calculated with the help of the transposed Jacobian

\[ \tau = J^T P \quad \text{Eq. 4.4} \]

where \( P \) is the force acting on the robot, consisting of the weight of the load and of the robot links, which are all directed in the negative Z-axis direction. The mass of each link is calculated using the link length and length density, with the weight applied at the middle of each link.

The dynamic work, \( E_d \), is calculated by

\[ E_d = \int_{q_i}^{q_f} I(q) \, \ddot{q} \, dq \quad \text{Eq. 4.5} \]

where \( I \) is the moment of inertia of the robot links and load, depending on the robot configuration, and \( \ddot{q} \) is the acceleration in the robot joints. It is assumed that the acceleration has a maximal value, \( \ddot{q}_{\text{max}} \), (bang-bang control), hence, it is taken as a constant. Finally, the dynamic work can be approximated as

\[ E_d = \ddot{q}_{\text{max}} \int_{q_i}^{q_f} I(q) \, dq \quad \text{Eq. 4.6} \]

The cost function \( F \) is therefore the average fruit-picking time, \( t \), for all picked fruit \( N_{\text{picked}} \)

\[ F = \left( \frac{\sum_{l=1}^{N_{\text{picked}}} t_{fr,l}}{N_{\text{picked}}} \right) \quad \text{Eq. 4.7} \]

The cost function \( F \) is measured in units of time (seconds). The time calculation is approximated and simplified, and does not include fruit-recognition time, trajectory-planning time, time for fruit detachment and placement in the gathering bin, etc. which are important, but do not depend on the kinematics of the robot. Therefore, the resultant time values, which are in the range [0.25 5] seconds in this research, are different from the values reported in previous studies, such as Van Henten et al. (2009), which are 45 seconds on average.
4.2. Optimization parameters

The optimization parameters are the known DH convention parameters $\alpha$, $\theta$, $a$ and $d$ (Craig, 2005). The total number of parameters defining the robot kinematics is $4 \times N_{DOF}$. The parameters representing the robot’s DOF (either $\theta$ or $d$) are found by solving the inverse kinematics for a specific robot configuration. Thus, the total number of free unconstrained optimization parameters is $3 \times N_{DOF}$.

Type and order of the robot joints can strongly affect the applicability of the robot structure to different environments. For a 3-DOF robot, we checked the following structures: RRR, RRP and PPP, where R and P represent revolute and prismatic (linear) joints, respectively, as shown in Fig. 4.1. These structures were chosen because they represent different types of robotic arms: RRR is an articulated robot, RRP is a telescopic robot, and PPP is a Cartesian robot.

Fig. 4.1 Robotic arms with 3 degrees of freedom of the robotic arm and different kinematic structures: PPP (a), RRP (b) and RRR (c). The arms are located on a platform vehicle, which is not considered part of the arm’s kinematics.
The location of the robot base constitutes an additional optimization parameter. Each location was defined by two parameters: $X$ and $Y$ coordinates on the ground plane. The search for the optimal base location was decoupled from the optimization of the robot kinematics, and was found for a given number of robot locations, $N_{loc}$, by the grid-search method with the branch and bound (BB) algorithm (Kolesar, 1967).

We also defined the limits of the optimization variables fitted to the task as follows:

- $\alpha$ parameter is in the interval $[-\pi, \pi]$.
- $\theta$ parameter is in the interval $[-\pi, \pi]$.
- $a$ and $d$ parameters are in the interval $[0, 3]$, considering that the height and width of orchard trees do not exceed 3 m.
- Similarly, the $X$ and $Y$ coordinates of the robot base location are in the interval $[-3, 3]$.

**4.3. Optimization constraint**

The constraint of the optimization is the robot's ability to perform its task in a given environment. The environment is a set of geometrical models of all targets and obstacles. Interaction with these objects influences the robot's motion: the robot must approach the targets without colliding with the obstacles.

To make the environment constraint more realistic, the allowed unpicked fruit percentage, $\alpha_{allow}$, is defined. This percentage depends on the economic aspects of harvesting. It is compared with the percentage of fruit that is not picked by the robot, $\alpha_{uf}$. Thus, the robot optimization constraint is $\alpha_{uf} \leq \alpha_{allow}$.

**4.4. Sub-problems of the robot optimization**

The robot optimization problem consists of a number of sub-problems. To evaluate the performance cost function, the inverse kinematics and trajectory planning problems have to be solved. To check the optimization constraint, robot collision with the environment has to be tested. For a given robot kinematics, the optimal robot location is found by the robot location optimization (described in Chapter 5). The parameters defining the robot kinematics are found by the robot kinematics optimization.
4.4.1. Robot kinematics optimization

The variables of this optimization are the parameters of the DH notation defining the kinematics of a robotic arm. The total number of optimization variables defining the robot kinematics for a 3-DOF robot is nine (Section 4.2). Since the number of variables is relatively large, only variables having a significant influence on robot effectiveness for the considered robot structures (PPP, RRP and RRR) were chosen. Some kinematics parameters for different robot structures were predefined, while the rest were the optimization variables or the robot DOF. This partition is presented in Table 4.1. The predefined parameters are presented by their numerical value. The optimization variables are marked in bold. The other three parameters were the robot DOF, which are defined by the inverse kinematics.

The Cartesian PPP structure has the prismatic DOF described by parameters $d_1$, $d_2$ and $d_3$. The optimization variables for the kinematics of this robot are $\theta_2$, which defines the angle of the last (third) link relative to the ground and $\alpha_2$. An additional optimization variable is the height of the robot base in the home configuration, which is not a part of the DH table.

The RRP structure has 2 revolute DOF described by the $\theta_1$ and $\theta_2$ parameters, and 1 prismatic DOF described by the $d_3$ parameter. The optimization variables for the kinematics are $d_1$ and $d_2$.

The RRR structure has 3 revolute DOF described by $\theta_1$, $\theta_2$, and $\theta_3$. The optimization variables for the kinematics are $d_1$, $a_2$ and $a_3$.

Table 4.1 *Optimization variables (bold) and constrained parameters defining the robot kinematics for the considered robot structures.*

<table>
<thead>
<tr>
<th></th>
<th>PPP</th>
<th>RRP</th>
<th>RRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\theta$</td>
<td>$\pi$</td>
<td>$\theta_2$</td>
<td>$\theta_1$</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>$\pi/2$</td>
<td>$a_2$</td>
<td>$a_1$</td>
</tr>
<tr>
<td>$a$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$d$</td>
<td>$d_1$</td>
<td>$d_2$</td>
<td>$d_3$</td>
</tr>
</tbody>
</table>
4.5. Methods for robot kinematics optimization

The optimization of robot kinematics is a complex computational problem. The following criteria are defined to test optimization methods for the needs of this research.

4.5.1. Criteria for the optimization method

1. The method has to be able to deal with 2 to 5 variables.

2. The evaluation of the cost function takes a relatively long time. Hence, the method has to have a good convergence rate.

3. The optimization problem cannot be formulated by continuous analytical expressions. Hence, the methods cannot be based on derivatives.

4. Because of the discontinuity, the cost function has numerous local minima. Hence, the method has to be able to search for the global minimum.

5. The optimization problem must be solved a large number of times to analyze the sensitivity on different parameters. Hence, the method has to be repeatable.

An example of a surface describing the cost function is shown in Fig. 4.2. The cost function is used to optimize a robot with the RRR structure for the TS apple tree model. To illustrate the cost function in 2D, only two optimization variables are changed: $a_2$ and $d_1$. The variables are taken from the domain [0, 1.5] m on the grid consisting of 50 vertices in each dimension. According to the top view in Fig. 4.2, the cost function is defined on a part of the domain. At the rest of the points the optimization constraint is not fulfilled. According to the front, side and isometric views, the function is non-smooth and has multiple local minima (marked by red).
Fig. 4.2 *Surface describing the cost function of a robot depending on variables* $a_2$ and $d_1$, $F(a_2,d_1)$. *The front (a), side (b), top (c) and isometric (d) views are shown. The colors of the surface represent the surface height proportional to the F value. The local minima are presented by red asterisks.*

The following methods were examined for the solution of the robot kinematics optimization problem: simplex Nelder–Mead, GA and grid search. All of these methods are able to solve problems with a large number of variables (fulfilling criterion 1), and do not use derivatives (criterion 3).
4.5.2. Simplex Nelder–Mead

Since the simplex method is deterministic, it is repeatable for a specific initial guess (criterion 5). The disadvantage of the simplex method is convergence to a local minimum close to the initial guess (criterion 4), which is critical for functions as in the previous example. To find the global minimum, different initial guesses have to be systematically checked. For example, the initial guesses can be arranged in a grid, which leads to the grid search method. The simplex method has a low convergence rate (criterion 2).

4.5.3. GA

The GA method is able to find the global minima for functions such as that in the example (criterion 4). The convergence rate (criterion 2) of a stochastic method is hard to estimate: the number of the function evaluations is predefined (population size multiplied by the number of generations), but the precision of the result cannot be neither expected nor measured. The repeatability (criterion 5) of the results is not sufficient for this study.

4.5.4. Grid search method

The number of function evaluations in the grid search method is defined by the resolution of the grid. This number can be relatively large, and therefore the convergence rate (criterion 2) of this method is low. Since the method checks all possibilities, it always finds the global minimum (criterion 4) with the defined resolution. Because of the predefined points in the grid, the method is convenient for comparing solutions with different conditions (criterion 5). The precision of the method is predefined by the resolution and is relatively rough in this study; nevertheless, it is sufficient for analysis and development of the methodology. While designing the robot, the precision can be improved by applying more precise methods (such as simplex and GA) using the result of the grid search method as the initial guess.

The main advantage of the grid search method is giving the overall picture of the cost function on a specific domain, which enables analysis of the cost function behavior.

4.6. Influence of the optimization constraint on the cost function

As it was described in section 4.3, the constraint of the optimization problem is the optimized robot's ability to reach a specific percentage of fruit on a tree. This constraint is defined by the
percentage of allowed unreachable fruit, $\alpha_{allow}$. The reason for defining $\alpha_{allow}$ is the assumption that there exist fruit representing the worst case in terms of the robot being able to reach them. Such fruit might be located furthest away from the robot base or, because of their orientation, can only be reached by a close to singular robot configuration. In both cases, the robot’s configurations have a maximal distance from the home configuration in the robot joint space, causing the maximal cost-function values. Such configurations are defined here as extreme configurations, and the fruit generating them are defined as extreme fruit.

There is a trade-off between the performance cost function $F$ and the amount of unreachable fruit $\alpha_{allow}$: the smaller the number of unpicked fruit, the more fruit with a high cost-function value that must be picked and therefore, the higher the total cost-function value, and vice versa. Hence, there exists an optimal value of $\alpha_{allow}$.

The optimal amount of allowed unreachable fruit can only be defined by considering all of the aspects of the robotic fruit harvesting, in particular economic aspects such as fruit cost and the cost of robot working time. Such an analysis was beyond the scope of this research, and we therefore have no specific reasonable value of $\alpha_{allow}$. Nevertheless, to learn about the described trade-off and help the robot designer find the optimal $\alpha_{allow}$, we performed a sensitivity analysis of the robot performance cost function $F$ depending on the allowed percentage of unreachable fruit $\alpha_{allow}$. An illustrative graph showing the dependence $F(\alpha_{allow})$ is presented in Example 3 (Section 4.8.3).

4.7. **Computational run time complexity for optimization problem solution**

As described in Section 4.4, the robot optimization problem has a number of nested sub problems, listed in the order of nesting: robot kinematics optimization, robot location optimization, robot motion planning and collision checking. The computational run time complexity of the problems and parameters used in this research are listed below.

The robot kinematics optimization is described in Section 4.4. Each parameter can achieve $n_{opt}$ values from some interval. Thus, the time complexity is $O(n_{opt}^3)$. Number of points, $n_{opt}$, in this research, is picked to be 6.
The robot location optimization problem is described in Section 5. The time complexity is evaluated in Section 5.3.3 and equals to \( O\left(\left(\frac{n_{\text{opt loc}}}{N_{\text{loc}}}\right)^{N_{\text{loc}}}\right)\), where \( n_{\text{opt loc}} \) is the number of possible points of the robot location, and \( N_{\text{loc}} \) is the number of location of the robot around the tree. We chose \( n_{\text{opt loc}} = 100 \) and \( N_{\text{loc}} = 2 \ldots 12 \), in this research.

The robot motion planning problem (together with the inverse kinematics problem) is solved for each fruit on the tree. Thus, the time complexity is \( O\left(N_{\text{fruit}}\right)\), where \( N_{\text{fruit}} \) is the number of fruit on the tree. \( N_{\text{fruit}} = 30 \ldots 1154 \) in this research.

The collision checking is checked during the motion planning for each tree branch. Thus the time complexity is \( O\left(N_{\text{branch}}\right)\), where \( N_{\text{branch}} \) is the number of branches on the tree. \( N_{\text{branch}} = 30 \ldots 2463 \) in this research.

The total computational run time complexity for the robot kinematics optimization problem is therefore

\[
O\left(n_{\text{opt}}^3 \cdot \left(\frac{n_{\text{opt loc}}}{N_{\text{loc}}}\right)^{N_{\text{loc}}} \cdot N_{\text{fruit}} \cdot N_{\text{branch}}\right). \quad \text{Eq. 4.8}
\]

Equation 4.8 does not take into account the number of iteration in the inverse kinematics solved by the Newton method and motion planning solved by the RRT method, which depends on the problem complexity. Thus, in the most complicated case considered in the research, the time complexity is at least \( O\left(6^3 \cdot \left(\frac{100}{12}\right)^{12} \cdot 1154 \cdot 2463\right) = O(10^{19})\).

**4.8. Examples of the agricultural task-based optimization**

The following examples present optimization of the robotic arm for fruit picking in specific environments: single trees and groups of trees. In the examples, the number of objects in the environment increases from the simplest case to the most complicated one. The simplest case's runtime is measured in hours; the more complicated the case, the longer the optimization runtime, eventually becoming unacceptably long (more than a week).

The calculation is performed using the MATLAB parallel computing tool on a computer with four cores (Intel® Core™ i5-2310 CPU 2.9GHz, 4GB RAM). This is sufficient to solve most of the optimization problems needed to describe the developed methodology in this research. Stronger
computers can be used to increase the number of problems that can be solved. Nevertheless, this increase is not significant: most of the actual problems may even be too complicated for supercomputers. The main goal of this research was to develop methods for analyzing the robot’s environment while decreasing optimization times to those feasible for regular computers.

The robot structures used in the examples are PPP, RRP and RRR. The maximal number of robot locations around a single tree is $N_{loc} = 8$ and along a tree row with five trees, $N_{loc} = 12$. The following trees and groups of trees are used as the robot environment: Tall Spindle (TS), Y-trellis (YT), and Central Leader (CT) apple tree models, and a row of five YT apple tree models.

4.8.1. Example 1: TS apple tree

The tree model includes 30 fruit and 30 branches (described in Appendix A.). Optimal robots with structures PPP, RRP and RRR for apple harvesting on the TS apple tree model are presented in Fig. 4.3 for two robot locations around the tree ($N_{loc} = 2$). The trajectories of the end effector are presented from the home configuration to the configurations for fruit picking. The performance of the optimal robots for different robot structures and numbers of locations is given in Table 4.2. The number of robots with different kinematics checked during the optimization is denoted by $N_{kin}$. The optimization runtime is given in hours in column T (Fig. 4.2).

![Fig. 4.3 Robots with structures PPP (a), RRP (b) and RRR (c) optimal for fruit picking on the model Tall Spindle apple tree with two optimal robot locations.](image)
Table 4.2 Performance cost function for the optimal robots with different structures and different numbers of locations (N_{loc}) around the tree for the Tall Spindle apple tree model.

<table>
<thead>
<tr>
<th>N_{loc}</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
<th>N_{kin}</th>
<th>T (h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPP</td>
<td>0.48</td>
<td>0.43</td>
<td>0.40</td>
<td>0.40</td>
<td>860</td>
<td>0.6</td>
</tr>
<tr>
<td>RRP</td>
<td>0.32</td>
<td>0.26</td>
<td>0.25</td>
<td>0.25</td>
<td>123</td>
<td>1.1</td>
</tr>
<tr>
<td>RRR</td>
<td>0.41</td>
<td>0.28</td>
<td>0.28</td>
<td>0.28</td>
<td>744</td>
<td>8.4</td>
</tr>
</tbody>
</table>

4.8.2. Example 2: YT apple tree

The tree model includes 67 fruit and 81 branches. Optimal robots with structures PPP, RRP and RRR for apple harvesting on the YT apple tree model are presented in Fig. 4.4 for four robot locations around the tree (N_{loc} = 4). The trajectories of the end effector are presented from the home configuration to the configurations for fruit picking. The performance of the optimal robots for different robot structures and numbers of locations is given in Table 4.3.

Fig. 4.4 Robots with the structures PPP (a), RRP (b) and RRR (c) optimal for fruit picking on the model Y-trellis apple tree with four optimal robot locations.
Table 4.3 Performance cost function for the optimal robots with different structures and different numbers of locations ($N_{loc}$) around the tree for the Y-trellis apple tree.

<table>
<thead>
<tr>
<th>$N_{loc}$</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
<th>$N_{kin}$</th>
<th>T (h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPP</td>
<td>1.9</td>
<td>1.1</td>
<td>0.95</td>
<td>0.9</td>
<td>728</td>
<td>3.6</td>
</tr>
<tr>
<td>RRP</td>
<td>2.4</td>
<td>1.6</td>
<td>1.45</td>
<td>1.45</td>
<td>26</td>
<td>0.6</td>
</tr>
<tr>
<td>RRR</td>
<td>1.24</td>
<td>0.84</td>
<td>0.81</td>
<td>0.81</td>
<td>140</td>
<td>1.6</td>
</tr>
</tbody>
</table>

4.8.3. Example 3: CL apple tree

The tree model includes 144 fruit and 171 branches. Optimal robots with structures PPP, RRP and RRR for apple harvesting on the CL apple tree model are presented in Fig. 4.5 for four robot locations around the tree ($N_{loc} = 4$). The trajectories of the end effector are presented from the home configuration to the configurations for fruit picking. The performance of the optimal robots for different robot structures and numbers of locations is given in Table 4.4.

Cells that include percentage of unpicked fruit, $\alpha_{uf}$ in Table 4.4 indicate that the optimal solution fulfilling the constraint $\alpha_{uf} < 5\%$ was not found. This happens when the number of robot locations around the tree is insufficient to reach all of the fruit on the tree.

![Fig. 4.5 Robots with the structures PPP (a), RRP (b) and RRR (c) optimal for fruit picking on the model Central Leader apple tree with four optimal robot locations.](image)
Table 4.4 Performance cost function for the optimal robots with different structures and different numbers of locations \( (N_{loc}) \) around the tree for the Central Leader apple tree.

<table>
<thead>
<tr>
<th>( N_{loc} )</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
<th>( N_{kin} )</th>
<th>( T ) (h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPP</td>
<td>1.1, ( \alpha_{uf} = 15% )</td>
<td>0.77, ( \alpha_{uf} = 13% )</td>
<td>1.97</td>
<td>1.87</td>
<td>790</td>
<td>79</td>
</tr>
<tr>
<td>RRP</td>
<td>1.3, ( \alpha_{uf} = 13% )</td>
<td>1.65, ( \alpha_{uf} = 8.3% )</td>
<td>1.13</td>
<td>1.12</td>
<td>123</td>
<td>10</td>
</tr>
<tr>
<td>RRR</td>
<td>1.35</td>
<td>0.87</td>
<td>0.71</td>
<td>0.61</td>
<td>790</td>
<td>521</td>
</tr>
</tbody>
</table>

In all environments and with all robot types, the cost-function value decreases as the number of locations around the tree, \( N_{loc} \), increases. The reason for this dependence is that the robot working volume is divided into smaller parts as the number of robot locations increases. As a consequence, the smaller the working volume, the shorter the lengths of the robot links and the smaller their masses; this results in a decrease in the energy and time required for their motion.

However, while the cost-function value decreases as \( N_{loc} \) increases up to 6, for \( N_{loc} = 8 \), the cost function barely changes. This can be observed in all robot types, and indicates that the larger number of robot locations has lower efficiency.

4.8.4. Sensitivity analysis

The sensitivity analysis of the function \( F(\alpha_{allow}) \) for the RRR robot working on the CL apple tree model is presented in Fig. 4.6. The four colors of the lines and points correspond to different numbers of robot locations \( (N_{loc}) \). Each point represents a robot with some kinematics considered during the grid search: cost-function value \( F \) of the robot and the percentage of unpicked fruit \( \alpha_{uf} \).

The percentage of unpicked fruit is checked in the interval from 0 to 15\%.

The optimal solution for each value of \( \alpha_{uf} \) is the point with the minimal cost-function value \( F \). These points are connected by lines for each number of robot locations. The lines seem non-smooth (compared with the same analysis for the simplified problem in Section 6.4.1) because of inaccuracies in the calculation of \( F \) caused by the rough resolution in the grid search. Nevertheless, the general behavior of the lines describes the obvious trade-off between \( F \) and \( \alpha_{allow} \): the more fruit that are allowed to be unpicked (higher \( \alpha_{allow} \)), the lower the cost function.
Analysis of this graph enables drawing conclusions about the value of the allowed unpicked fruit, \( \alpha_{\text{allow}} \). For example, for \( N_{\text{loc}} = 6 \), the robot picking all of the fruit, \( \alpha_{\text{allow}} = 0 \), has \( F = 0.83 \), while a robot picking 95\% of the fruit, \( \alpha_{\text{allow}} = 5\% \), has \( F = 0.73 \), which is 12\% lower. Thus, comparing the cost of the unpicked fruit and the robot working time, the design can find the optimal value of the allowed unpicked fruit \( \alpha_{\text{allow}} \).

**Example 4: group of five YT apple trees**

The tree models include 297 fruit and 379 branches (shown in Fig. 4.7 a). The number of objects and the size of this environment are too large to determine the optimal robot in an acceptable time. Checking 12 robot locations (\( N_{\text{loc}} = 12 \)) leads to the combinatorial explosion of robot location combinations, causing the robot location optimization problem to take up the main portion of the solution time. For all robot structures, a single evaluation of the optimization cost function requires more than a day.

As an example, one non-optimal robot with RRR structure was found. The calculation to find it took 3 days. That robot is presented in Fig. 4.7 b, and its cost function, \( F = 1.6 \), was almost twice that found with the method developed in the research (Section 8.5).
Fig. 4.7 Model of Y-trellis apple row with five trees is too complicated to find optimal robots (a). Non-optimal robot performing the picking task for the Y-trellis apple row (b).

4.9. Comparison to industrial robots with suitable kinematics

A number of industrial articulated robots with kinematic parameters suitable for the considered fruit-harvesting tasks were compared with the achieved optimal robots. Since the industrial robots usually have 6 DOF, the last 3 DOF corresponding to the wrists (typically influencing only the orientation of the robot end effector) are fixed to compare these robots with the optimized robots with 3 DOF. Thus, only the RRR robot structure was compared.

To compare the effectiveness of the robot kinematics, the dynamic parameters of the industrial robots, such as joint angular velocity, were calculated similar to the optimized robot, and were not taken from the specifications of the industrial robot. The robot with kinematics taken from the industrial robot was located in the optimal locations around the tree and the optimal height of the robot base was found.

The following industrial robots were considered: FANUC P-250iA/15, FANUC P-250iA/10S, KUKA KR 16 L6-2 and Yaskawa Motoman HP20D-A80. Their kinematics are shown in Fig. 4.8 in the sketches taken from the robot specifications.
Fig. 4.8 Robot kinematics of the FANUC P-250iA/15 (a), FANUC P-250iA/10S (b), KUKA KR 16 L6-2 (c) and Yaskawa Motoman HP20D-A80 (d) models (taken from the robot specifications of FANUC, KUKA and Yaskawa companies).

The performance cost functions of the optimized robots were compared with those of the robots with kinematics taken from the considered industrial robots. The results of the comparison for the
YT and CL apple tree models are presented in Table 4.5, where the best value in each row is in bold. The empty cells represent cases in which the robot did not satisfy the optimization constraint.

Most of the considered industrial robots had larger cost-function values than the optimized robots. Nevertheless, the cost function of the robot with the kinematics of the Motoman model was similar to or less than that of the optimized robot. This kinematics was not found during the optimization because of the roughness of the grid in the optimization algorithm. Thus, if we were to use an industrial robot for fruit harvesting, the most fitted robot model, including its optimal locations around the tree and the height of the robot base, could be found by the task-based optimization. Nonetheless, checking all the kinematics of the existing industrial robots cannot provide the actual optimal kinematics which can be derived only by the optimization.

Together with the high importance of the agricultural robot to be optimized, the complexity of the agricultural environment is also high, decreasing the ability to fit the robot to the environment. To break this circle, more effective methods for solving the optimization problem were developed in this study (Chapters 6 and 7).

Table 4.5 Comparison of optimized robots kinematics and kinematics taken from industrial robots.

<table>
<thead>
<tr>
<th>Robot</th>
<th>Y-trellis</th>
<th>Central Leader</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$N_{loc}$</td>
<td>$N_{loc}$</td>
</tr>
<tr>
<td></td>
<td>2 4 6 8</td>
<td>2 4 6 8</td>
</tr>
<tr>
<td>Optimized</td>
<td>1.04 0.81 0.78 0.81</td>
<td>1.35 0.87 0.71 0.68</td>
</tr>
<tr>
<td>FANUC P-250iA/15</td>
<td>2.88 2.25 1.88 1.87</td>
<td>2.61 2.53 1.65 1.58</td>
</tr>
<tr>
<td>FANUC P-250iA/10S</td>
<td>1.37 1.34 1.15 1.13</td>
<td>1.28 1.23</td>
</tr>
<tr>
<td>KUKA KR 16 L6-2</td>
<td>1.65 1.4 1.08 1.01</td>
<td>1.59 0.99 0.82 0.82</td>
</tr>
<tr>
<td>Motoman HP20D-A80</td>
<td><strong>0.85</strong></td>
<td><strong>0.89</strong></td>
</tr>
</tbody>
</table>
5. Optimal Robot Location Problem

Defining the optimal location of a robot is a critical stage of the robot optimization. In the industrial domain, this problem is relatively simple due to the structured environment, low number of targets and the fact that, for most industrial tasks, a single robot location is sufficient. None of these conditions are valid in the agricultural domain. Since an entire tree or entire orchard cannot be harvested by a robot located in a single location, the robot has to perform this task in multiple locations. In this research, we assumed that the problem of optimal location search for a single robot in a number of locations is equivalent to the problem of optimal location search for a number of identical not cooperative robots.

5.1. Definition of optimal location problem

The general robot optimization problem consists of optimization of the robot kinematics and robot location. To simplify the general problem, it is separated into two nested problems: optimization of the robot kinematics (checking all possible robot kinematics) and optimization of the robot location (for each robot kinematics checking all possible robot locations). Then the optimal location problem becomes the optimization problem with cost function and constraints similar to those for the robot kinematics optimization defined in Section 4.1, 4.2 and 4.3.

The cost function $F$ is the average fruit-picking time. The constraint is picking a specific percentage of fruit, $\alpha_{of} < \alpha_{allow}$, where $\alpha_{of}$ is the percentage of unpicked fruit, and $\alpha_{allow}$ is the percentage of allowed unpicked fruit. The optimization parameters are robot location and orientation, i.e., coordinates $X$ and $Y$ of the robot base and orientation $\Theta$ in the tree coordinate system, which is sufficient to describe the robot position around the tree. However, robots with a first revolving joint, such as RRR or RRP, can change their orientation, and therefore the orientation parameter $\Theta$ is part of the inverse kinematics problem and is omitted in the solution of the optimal location problem.

An additional parameter of the optimal location problem is the number of locations, $N_{loc}$. We assume that the robotic arm performing the harvesting is mounted on a platform vehicle moving inside the orchard. The motion of the platform is performed in some specific time, hence, the total harvesting time consists of the time it takes for the robotic arm to move to the fruit, and the time
taken by the platform to move between robot locations. The time of the platform motion is defined by the technical specifications of the platform and must be decided in each specific case. To simplify the optimization problem and not include the unknown time of the platform motion, we do not use it during the solution.

To model a realistic case, where the robot picks fruit from both sides of the tree row, the considered $N_{\text{loc}}$ is taken as even for equal partitioning between the sides. The minimal $N_{\text{loc}}$ is 2. The maximal $N_{\text{loc}}$ considered in this research is 12.

Finally, the optimal location problem is defined as follows: for given robot kinematics (parameters of the DH table) and number of locations, $N_{\text{loc}}$, minimize

$$
\min F(a, d, \alpha, \theta, X, Y)
$$

for $X_i, Y_i, i = 1..N_{\text{loc}}$ \hspace{1cm} Eq. 5.1

s.t. $\alpha_{\text{uf}} \leq \alpha_{\text{allow}}$

for a robot with first revolute joint, and

$$
\min F(a, d, \alpha, \theta, X, Y, \Theta)
$$

for $X_i, Y_i, \Theta_i, i = 1..N_{\text{loc}}$ \hspace{1cm} Eq. 5.2

s.t. $\alpha_{\text{uf}} \leq \alpha_{\text{allow}}$

for a robot with first prismatic joint.

5.2. Method for solving the optimal location problem

The number of optimization parameters in the robot location optimization problem is $2 \times N_{\text{loc}}$ or $3 \times N_{\text{loc}}$. In the worst case, when $N_{\text{loc}} = 12$, it is $3 \times 12 = 36$ parameters. Similar to the robot kinematics optimization problem (Section 4.5), an appropriate method for solving this problem should be found. The criteria for the method are the same as those listed in Section 4.5 for the robot kinematics optimization.

1. The optimization problem has a relatively large number of variables. Here, the method has to be able to deal with up to 36 variables.
2. The evaluation of the cost function takes a relatively long time. Hence, the method has to have a good convergence rate.

3. The optimization problem cannot be formulated by continuous analytical expressions. Hence, the methods cannot be based on derivatives.

4. Because of the discontinuity, the cost function has numerous local minima. Hence, the method has to be able to search for the global minimum.

5. In this research, the optimization problem must be solved a large number of times with same and different parameters. Hence, the method has to be repeatable.

Taking into account the discontinuity of the performance cost function (when a small change in robot location prevents it from reaching some fruit, the cost function changes value sharply), it is impractical to solve this problem by methods intended for continuous problems.

To solve this problem with the grid search method, it was defined in a discrete formulation. A number of possible robot locations were defined in the nearest area around a tree. Typically, about 100 points of possible robot locations were distributed uniformly in a round area with a radius of 2 m and the tree trunk at the center. From each location, the cost function of the harvesting of each fruit was calculated and listed in a table. Then the solution of the optimal robot location problem was achieved by finding the optimal combination of possible robot locations covering the needed number of fruit with the minimal total cost function. This problem was a combinatorial problem, solved by checking all combinations of possible robot locations.

Three methods were examined: derivative free deterministic simplex method, stochastic GA method and brute force grid search method. To compare the performance of the examined methods and to find the most suitable method, an example of the robot location optimization was considered. The robot had a PPP structure, and four optimal robot locations around the TS tree model. The four optimal locations were determined in the different methods examined.

5.2.1. Grid search method

The main advantage of the grid search method is that it gives an overall picture of the cost function on a specific domain. This enables analysis of cost function behavior. The grid search method is deterministic (criterion 5), and is able to deal with a large number of variables (criterion
1) without smooth analytical expressions (criterion 3) and with the ability to search for the global minimum (criterion 4). The convergence rate of the method (criterion 2) depends on the required resolution. For a large number of variables, the number of function evaluations can be unacceptably large. Nevertheless, using improvement algorithms, such as the BB method, the grid search can be the most suitable method for the robot location optimization solution in this research. The improvements and required assumptions are described in this chapter.

Despite the fact that the simplex and GA methods found solutions with better cost functions than those found by the grid search method, the solutions of the simplex and GA methods are based on the initial guesses found by the grid search method. This means that the grid search method must be applied anyway, and the results can be improved by more precise methods.

5.2.2. Simplex Nelder–Mead method

The simplex method is deterministic (criterion 5), and is able to deal with a large number of variables (criterion 1) without smooth analytical expressions (criterion 3). Nevertheless, the convergence rate of the method is low (criterion 2). Because of the discontinuity of the cost function, there are a large number of unsuccessful initial guesses with undefined function values. Hence, to find the global minimum (criterion 4), numerous initial guesses have to be systematically checked. The initial guesses can be arranged in a grid, turning the search of initial guesses into the grid search method.

5.2.3. GA

The stochastic method GA is able to deal with a large number of variables (criterion 1) without smooth analytical expressions (criterion 3) and with the ability to search for the global minimum (criterion 4). Nevertheless, the convergence rate of the method is low (criterion 2). The repeatability of the method (criterion 5) is unacceptable: the solutions for different runs of the method give completely different locations of the robot around the tree and significantly different cost-function values.
5.2.4. Evaluation of the methods

The close-to-optimal solution achieved by the grid search method was the four robot locations shown in Fig. 5.1a by asterisks, and it was taken as a reference for the comparison. The cost-function value found by the grid search method was $F = 0.45$.

Twenty random initial guesses for the simplex method were created based on the grid search solution: all of the initial guesses had a random deviation of their coordinates from the grid search solution, with maximal deviation of 0.1 m. The near-optimal locations achieved by the simplex method based on the random initial guesses are shown in Fig. 5.1a as different-colored dots (four dots per each color represent one solution). These solutions represent local minima of the optimization cost function found by the simplex method. The distribution of the cost-function values found by the simplex method from the random initial guesses is shown in the histogram in Fig. 5.1b. All of the solutions were better than the solution found by the grid search method, although to find them, the grid search had to be performed first.

![Fig. 5.1 Close-to-optimal robot locations achieved by the simplex method with 20 random initial guesses presented by dots in different colors. The near-optimal robot locations achieved by the grid search method are shown as asterisks (a). The distribution of the cost-function values for the found solution is presented in the histogram (b).](image-url)
Fig. 5.2 Close-to-optimal robot locations achieved by 20 runs of the genetic algorithm. The near-optimal robot locations achieved by the grid search method are shown as asterisks (a). The distribution of the cost-function values for the found solution is presented in the histogram (b).

Twenty runs of the GA with a population of 200, 50 generations and a mutation rate of 0.2 were performed to find the near-optimal robot location combination. Similar to the initial guesses for the simplex method, the initial population was created as a deviation of the optimal locations achieved by the grid search method, with maximal deviation of 0.1 m. The near-optimal locations achieved by the GA method are shown in Fig. 5.2a as different-colored dots. These solutions represent local minima of the optimization cost function found by the GA method. The distribution of the cost-function values found by the GA method is shown in the histogram in Fig. 5.2b. Similar to the simplex method, all of the solutions were better than the solution found by the grid search method, although to find them, the grid search had to be performed first.

5.3. Solution by the grid search method

The grid search method was found to be suitable for the solution of the optimal robot location problem. To be solved by the grid search method, the problem must be formulated in a discrete way, i.e., all of the optimization variable values are taken from a finite predefined set constituting the search grid.

5.3.1. Possible robot locations

The resolution of the grid, \( r_{grid} \), is defined for each tree, depending on the tree's size. The grid consists of points on the ground plane around the tree representing the possible robot locations. If
the robot structure cannot change its orientation (as with the PPP), the additional dimension including the robot orientation is added to the grid.

The grid resolution is defined as the distance between two closest locations. The number of locations cannot exceed some limit to prevent a combinatorial explosion of possibilities. Thus, the resolution of the grid is chosen such that the number of vertices, \( N_{\text{ver}} \), does not exceed 100.

For a single tree, the possible robot locations are distributed around the tree in concentric circles with the tree trunk at their center. An example of a grid consisting of possible robot locations around a tree is given in Fig. 5.3. The tree model is the TS apple tree. The resolution providing \( N_{\text{ver}} = 104 \) possible robot locations is \( r_{\text{grid}} = 0.14 \) m. The nearest (\( D_{\text{min}} \)) and farthest (\( D_{\text{max}} \)) distances of the robot locations are defined according to the tree dimensions:

\[
D_{\text{min}} = \frac{R_{\text{max}}}{4}, \quad D_{\text{max}} = \frac{R_{\text{max}} + (Z_{\text{max}} - Z_{\text{min}})}{4}
\]

Eq. 5.3

where \( R_{\text{max}} \) is the maximal distance of fruit from the tree trunk, and \( Z_{\text{max}} - Z_{\text{min}} \) is the interval including the fruit from the lowest fruit height \( Z_{\text{min}} \) up to the highest fruit height \( Z_{\text{max}} \).

Usually, trees arranged in a row along the \( Y \) axis have no gap between them, or the gap is too small for a robot location. Hence, the robot locations inside the sectors shown in Fig. 5.3 are not considered. The angle of the sectors is taken as 60°.

Fig. 5.3 Possible robot locations around a tree distributed concentrically and represented by dots. Robot location inside the sectors with open angle of 60° is assumed to be impossible.
Fig. 5.4 Possible robot locations along the tree row, represented by dots.

For a row of trees, the possible robot locations are distributed along the tree. An example of a grid along the tree row is given in Fig. 5.4. The tree model is a row of YT apple trees. The resolution providing \( N_{\text{ver}} = 100 \) possible robot locations is \( r_{\text{grid}} = 0.54 \text{ m} \). The nearest \((D_{\text{min}})\) and farthest \((D_{\text{max}})\) distances of the robot locations are calculated as before.

5.3.2. Fruit–location map

To solve the optimal location problem, the values of the cost function \( F \) for all fruit picked from all robot locations are needed. The cost-function values for each fruit–location combination are set in a table, defined as the fruit–location map. It consists of \( N_{\text{fruit}} \) columns and \( N_{\text{ver}} \) rows. Each table cell \((i,j)\) has the cost-function value for the picking of fruit \( i \) from location \( j \) by the given robot. If the picking is impossible, the cell has infinity, or a large number \((10^6\) in this case).

As an example, the fruit–location map for the TS apple tree model and RRP robot is considered. A fragment of the fruit–location map is given in Table 5.1. The table presenting the fruit–location map consists of 100 rows and 30 columns. Thus, the cost-function value \( F \) for picking the third fruit from the fourth location is 0.43. Picking the aforementioned third fruit from the third location is impossible.
Table 5.1 *Fragment of the fruit–location map for the TS apple tree model and RRP robot.*

<table>
<thead>
<tr>
<th>fruit i location j</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>0.16</td>
<td>0.16</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>106</td>
<td>106</td>
<td>0.29</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>0.18</td>
<td>0.07</td>
<td>0.06</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>0.07</td>
<td>0.06</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>106</td>
<td>106</td>
<td>0.43</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>0.10</td>
<td>0.09</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.53</td>
<td>106</td>
<td>0.29</td>
<td>0.40</td>
<td>106</td>
<td>106</td>
<td>0.45</td>
<td>0.30</td>
<td>0.10</td>
<td>0.09</td>
</tr>
<tr>
<td>6</td>
<td>0.51</td>
<td>106</td>
<td>0.28</td>
<td>0.38</td>
<td>106</td>
<td>106</td>
<td>0.36</td>
<td>0.11</td>
<td>0.09</td>
<td>0.07</td>
</tr>
<tr>
<td>7</td>
<td>0.43</td>
<td>106</td>
<td>0.29</td>
<td>0.26</td>
<td>106</td>
<td>106</td>
<td>0.18</td>
<td>0.13</td>
<td>0.09</td>
<td>0.08</td>
</tr>
<tr>
<td>8</td>
<td>0.41</td>
<td>106</td>
<td>0.27</td>
<td>0.24</td>
<td>106</td>
<td>106</td>
<td>0.16</td>
<td>0.11</td>
<td>0.09</td>
<td>0.07</td>
</tr>
<tr>
<td>9</td>
<td>106</td>
<td>106</td>
<td>0.27</td>
<td>0.25</td>
<td>106</td>
<td>106</td>
<td>0.11</td>
<td>0.08</td>
<td>0.06</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.71</td>
<td>106</td>
<td>0.32</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>0.14</td>
<td>0.44</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>0.43</td>
<td>0.64</td>
<td>0.32</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>0.13</td>
<td>0.11</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>0.42</td>
<td>0.64</td>
<td>0.31</td>
<td>0.54</td>
<td>106</td>
<td>106</td>
<td>0.22</td>
<td>0.12</td>
<td>0.11</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>0.44</td>
<td>0.62</td>
<td>0.31</td>
<td>0.52</td>
<td>106</td>
<td>106</td>
<td>0.61</td>
<td>0.12</td>
<td>0.10</td>
<td></td>
</tr>
</tbody>
</table>

The solution of the optimal location problem (with given resolution), having the fruit–location map, is trivial for the single robot location, $N_{loc} = 1$. In this case, the optimal location corresponds to the row in the fruit–location map having less than the allowed non-infinite values (representing the unpicked fruit, meaning the fulfillment of the optimization constraint), and the minimal sum of non-infinite values (meaning minimization of the total or average picking time).

In the case of $N_{loc} > 1$, the fruit–location map cannot be obtained without additional computations. For example, if $N_{loc} = 2$, the optimal combination of two different robot locations must be found. Thus, all possible pairs of location combinations must be checked.

Taking the map from the previous example, the combination of the two following locations is checked: the fifth location (row 5 in the fruit–location map), denoted as $r_5$, and the eleventh location, denoted as $r_{11}$. The two rows are shown together in Table 5.2.

64
Table 5.2 *Fragment of the fruit–location map.*

<table>
<thead>
<tr>
<th>fruit i location j</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.53</td>
<td>106</td>
<td>0.29</td>
<td>0.40</td>
<td>106</td>
<td>106</td>
<td>0.45</td>
<td>0.30</td>
<td>0.10</td>
<td>0.08</td>
</tr>
<tr>
<td>11</td>
<td>0.43</td>
<td>0.64</td>
<td>0.32</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>0.13</td>
<td>0.11</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.3 *Fragment of the fruit–location map with location combinations.*

<table>
<thead>
<tr>
<th>fruit i location comb.</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>(5,11)</td>
<td>0.43</td>
<td>0.64</td>
<td>0.29</td>
<td>0.40</td>
<td>106</td>
<td>106</td>
<td>0.45</td>
<td>0.30</td>
<td>0.10</td>
<td>0.08</td>
</tr>
</tbody>
</table>

To distribute the fruit optimally between the robot locations, the minimal cost-function values must be taken for each fruit. For example, the first fruit must be picked from location 11, since the picking cost function has a lower value than that for location 5. Choosing the lowest cost-function values for two robot locations, $r_5$ and $r_{11}$, a new row describing the location combination $r_{(5,11)} = \min(r_5,r_{11})$ can be added to the fruit–location map (0). Now this row can be compared with the other rows, and an optimal row can be found in the same way as for $N_{loc} = 1$.

We construct a fruit–location map for each $N_{loc}$. Having the fruit–location map for $N_{loc} = 1$, we can compute the fruit–location map for $N_{loc} = 2$ by checking all of the combinations of rows of the fruit–location map for $N_{loc} = 1$. The total number of location combinations for $N_{loc}$ is $\binom{N_{loc}}{N_{per}}$.

An example of a fragment of a fruit–location map for $N_{loc} = 2$ with the list of the location combinations is shown in Table 5.4.
Table 5.4 Fragment of the fruit–location map with combination of two locations ($N_{\text{loc}} = 2$). The numbers of the locations in the combinations are given in the far-right columns.

<table>
<thead>
<tr>
<th>fruit i comb. j,k</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>locations j</th>
<th>k</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.06</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>0.75</td>
<td>106</td>
<td>0.68</td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1.06</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>0.75</td>
<td>106</td>
<td>0.68</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1.06</td>
<td>106</td>
<td>1.22</td>
<td>106</td>
<td>106</td>
<td>1.59</td>
<td>0.75</td>
<td>106</td>
<td>0.68</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>1.06</td>
<td>106</td>
<td>106</td>
<td>1.48</td>
<td>1.30</td>
<td>0.75</td>
<td>106</td>
<td>0.68</td>
<td>6</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1.06</td>
<td>106</td>
<td>1.25</td>
<td>106</td>
<td>106</td>
<td>0.75</td>
<td>106</td>
<td>0.68</td>
<td>7</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1.06</td>
<td>106</td>
<td>1.23</td>
<td>106</td>
<td>106</td>
<td>0.75</td>
<td>106</td>
<td>0.68</td>
<td>8</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1.06</td>
<td>106</td>
<td>1.31</td>
<td>106</td>
<td>106</td>
<td>1.58</td>
<td>0.75</td>
<td>1.43</td>
<td>0.68</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>1.06</td>
<td>106</td>
<td>1.23</td>
<td>106</td>
<td>1.47</td>
<td>1.56</td>
<td>0.75</td>
<td>106</td>
<td>0.68</td>
<td>13</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>1.06</td>
<td>106</td>
<td>1.25</td>
<td>106</td>
<td>1.39</td>
<td>0.75</td>
<td>106</td>
<td>0.68</td>
<td>14</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1.06</td>
<td>106</td>
<td>1.30</td>
<td>106</td>
<td>106</td>
<td>0.75</td>
<td>106</td>
<td>0.68</td>
<td>15</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1.06</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>0.75</td>
<td>106</td>
<td>0.68</td>
<td>16</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>1.06</td>
<td>106</td>
<td>1.47</td>
<td>106</td>
<td>1.80</td>
<td>1.68</td>
<td>0.75</td>
<td>106</td>
<td>0.68</td>
<td>21</td>
<td>3</td>
</tr>
<tr>
<td>13</td>
<td>1.06</td>
<td>106</td>
<td>1.48</td>
<td>106</td>
<td>1.85</td>
<td>1.82</td>
<td>0.75</td>
<td>106</td>
<td>0.68</td>
<td>22</td>
<td>3</td>
</tr>
</tbody>
</table>

In the same way, checking all possible location combinations including $N_{\text{loc}}$ locations, the fruit–location map for $N_{\text{loc}}$ can be constructed. This map has $C_{N_{\text{loc}}}^{N_{\text{loc}}}$ rows. In principle, with the help of the map, the optimal locations can be found for any $N_{\text{loc}}$ (with given resolution). Nevertheless, the number of location combinations can reach high values. In the example with the grid consisting of 100 possible robot locations and a search for eight optimal locations, the number of location combinations is $C_{100}^8 = \frac{100!}{(100-8)!8!} \approx 1.8 \cdot 10^{11}$. 

66
5.3.3. Decreasing the number of location combinations

To decrease the number of location combinations and make the optimal robot location problem solvable in an acceptable time, the four following methods were used: dividing the locations into sectors, applying the BB method, using bounding tolerance, and dividing the locations into two groups.

Dividing the locations into sectors

To decrease the number of location combinations, we assumed that the optimal robot locations cannot be close to each other. If they were close, the robots located in them could reach almost the same fruit in almost the same time. Thus, one of the locations should be canceled or moved to a more distant location.

According to this assumption, the close optimal robot locations are inefficient and are not taken into consideration. To spread the optimal robot locations around the tree, they are sought in equal sectors around the tree. The number of sectors is taken to be equal to $N_{loc}$.

Divisions into 2, 4 and 6 sectors for the TS apple tree model are shown in Fig. 5.5. Now, to find the combinations of optimal locations, only locations from the different sectors are taken. For example, for six optimal robot locations ($N_{loc} = 6$), the first optimal location is sought within the locations in the first sector (locations 1, 2, 6–8, 13–15, 23–26, 36–40), the second optimal location is sought in the second sector, and so on.

This partition into sectors significantly decreases the total number of possible location combinations. Taking the previous example for the TS apple tree model with 100 possible locations and eight optimal locations, the possible robot locations were divided into eight sectors. Each sector has about $\lceil 100/8 \rceil = 13$ locations. Then the number of location combinations is

$$ \left( \left\lfloor \frac{100}{N_{loc}} \right\rfloor \right)^{N_{loc}} = 13 \cdot 13 \cdot 13 \ldots \cdot 13 = 13^8 \approx 8.1 \cdot 10^8, $$

which is smaller by three orders of magnitude than when the combinations included any possible location.
Applying the BB method

The BB method is a well-known improvement for algorithms solving combinatorial optimization problems (Land and Doig, 1960). The rule of the method is as follows: if during the optimal location combination search, some combination is known to be worse than the others and all combinations derived from this combination will be worse, this combination with all of its derived ones will not be considered in the further search. Such a location, excluded from further consideration, is called a “discarded location”.

The BB method is applied during the creation of the location combinations for different $N_{loc}$. It can already be useful for $N_{loc} = 1$. For example, locations 7 and 10 are taken from the fruit–location map and are presented fully in Table 5.5. All values in row 7 are lower than the values in row 10. This means that location 10 is worse than location 7: all fruit that can be picked from location 10 can be picked from location 7 as well, but at lower cost. Therefore, location 10 can be eliminated from the fruit–location map, decreasing the number of locations and, as a result, the number of location combinations.
Table 5.5  Fragment of the fruit–location map for the Tall Spindle apple tree and RRP robot including the costs for fruit picking from locations 7 and 10.

<table>
<thead>
<tr>
<th>fruit i location j</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>0.43</td>
<td>106</td>
<td>0.29</td>
<td>0.26</td>
<td>106</td>
<td>106</td>
<td>0.18</td>
<td>0.13</td>
<td>0.09</td>
<td>0.08</td>
</tr>
<tr>
<td>10</td>
<td>0.71</td>
<td>106</td>
<td>0.32</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>0.17</td>
<td>0.44</td>
</tr>
<tr>
<td>fruit i location j</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
</tr>
<tr>
<td>7</td>
<td>0.12</td>
<td>0.11</td>
<td>0.14</td>
<td>0.14</td>
<td>0.10</td>
<td>0.05</td>
<td>0.06</td>
<td>106</td>
<td>106</td>
<td>106</td>
</tr>
<tr>
<td>10</td>
<td>106</td>
<td>0.32</td>
<td>0.14</td>
<td>0.14</td>
<td>106</td>
<td>0.08</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>106</td>
</tr>
<tr>
<td>fruit i location j</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
</tr>
<tr>
<td>7</td>
<td>0.24</td>
<td>0.05</td>
<td>0.06</td>
<td>0.18</td>
<td>0.03</td>
<td>106</td>
<td>0.02</td>
<td>106</td>
<td>106</td>
<td>106</td>
</tr>
<tr>
<td>10</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>106</td>
</tr>
</tbody>
</table>

This row elimination is performed when the fruit–location map is generated for each $N_{loc}$ from 1 to 8. The number of location combinations achieved by the search with the BB method will be presented in Table 5.7. These numbers are compared with the theoretical number of combinations (after division into sectors).

**Bounding tolerance**

Cases similar to that just described, where one location is better than another, are rare. Usually, when two locations are compared, one of the locations (called first) has a lower cost-function value for most of the fruit, but there is a relatively small number of fruit having a lower cost-function value at the second location, for example, locations 18 and 19 presented in Table 5.6. Location 19 has a lower cost function for most of the fruit. Location 18 has a lower cost function for fruit 13, 14 and 18, but this advantage is relatively small. In addition, all of the fruit that can be picked from location 18 can also be picked from location 19. Thus, in general, location 19 is assumed to be better than location 18, and, consequently, location 18 is discarded from the further optimal location search.
Table 5.6 Fragment of the fruit–location map with two locations, where the cost-function values for location 19 are lower than those for location 18 for most of the fruit.

<table>
<thead>
<tr>
<th>fruit i</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>location j</td>
<td>18</td>
<td>0.36</td>
<td>0.32</td>
<td>0.25</td>
<td>0.23</td>
<td>106</td>
<td>0.18</td>
<td>0.16</td>
<td>106</td>
<td>0.08</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>0.35</td>
<td>0.32</td>
<td>0.24</td>
<td>0.22</td>
<td>0.17</td>
<td>0.17</td>
<td>0.15</td>
<td>0.13</td>
<td>0.08</td>
</tr>
<tr>
<td>fruit i</td>
<td>11</td>
<td>0.10</td>
<td>0.10</td>
<td>0.11</td>
<td>0.11</td>
<td>0.09</td>
<td>0.06</td>
<td>0.06</td>
<td>0.06</td>
<td>0.07</td>
</tr>
<tr>
<td>location j</td>
<td>18</td>
<td>0.10</td>
<td>0.10</td>
<td>0.12</td>
<td>0.12</td>
<td>0.09</td>
<td>0.06</td>
<td>0.06</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>0.10</td>
<td>0.10</td>
<td>0.12</td>
<td>0.12</td>
<td>0.09</td>
<td>0.06</td>
<td>0.06</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>fruit i</td>
<td>21</td>
<td>0.14</td>
<td>0.06</td>
<td>0.06</td>
<td>0.10</td>
<td>0.05</td>
<td>106</td>
<td>0.10</td>
<td>106</td>
<td>0.03</td>
</tr>
<tr>
<td>location j</td>
<td>18</td>
<td>0.06</td>
<td>0.05</td>
<td>0.06</td>
<td>0.09</td>
<td>0.04</td>
<td>0.13</td>
<td>0.03</td>
<td>106</td>
<td>0.04</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>0.06</td>
<td>0.05</td>
<td>0.06</td>
<td>0.09</td>
<td>0.04</td>
<td>0.13</td>
<td>0.03</td>
<td>106</td>
<td>0.04</td>
</tr>
</tbody>
</table>

To discard a location that is not strictly better than another location (its cost function is not lower for all of the fruit), the bounding tolerance $t_b$ is defined. If two locations are compared, and the cost-function values of one of them (called first) are lower than the cost-function values of the other (called second), or are only higher than those of the second location by the $t_b$, the second location is discarded from further search. In the example with locations 18 and 19, approaching all of the fruit except 13, 14 and 18 has a lower cost from location 19. Approaching fruit 13, 14 and 18 from location 18 costs 9% less than the cost of approaching from location 19. Therefore, it is assumed that location 18 has no significant advantage and is discarded.

In this research, the bounding tolerance is taken as $t_b = 10\%$. The number of location combinations achieved by a search with the BB algorithm with $t_b = 10\%$ is presented in Table 5.7. In this case, the location optimization with the $t_b = 10\%$ gives a solution equal to that with location optimization without bounding tolerance ($t_b = 0$).
Table 5.7 Comparison of the number of location combinations for the complete search, search after division into sectors, search with the branch and bound (BB) algorithm without bounding tolerance, and search with the BB algorithm with bounding tolerance $t_b = 10\%$.

<table>
<thead>
<tr>
<th>$N_{loc}$</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Theoretical number of combinations</td>
<td>100</td>
<td>$C_{100}^2 \approx 10^4$</td>
<td>$C_{100}^4 \approx 10^8$</td>
<td>$C_{100}^6 \approx 10^{12}$</td>
<td>$C_{100}^8 \approx 10^{16}$</td>
</tr>
<tr>
<td>Number of combinations after dividing into sectors</td>
<td>100</td>
<td>$(\frac{100}{2})^2 = 2500$</td>
<td>$(\frac{100}{4})^4 = 390625$</td>
<td>$(\frac{100}{6})^6 \approx 24\cdot10^6$</td>
<td>$(\frac{100}{8})^8 \approx 815\cdot10^6$</td>
</tr>
<tr>
<td>Number of combinations with BB</td>
<td>53</td>
<td>95</td>
<td>8311</td>
<td>29438</td>
<td>42036</td>
</tr>
<tr>
<td>Number of combinations with BB and $t_b = 10%$</td>
<td>38</td>
<td>61</td>
<td>1502</td>
<td>3618</td>
<td>3936</td>
</tr>
</tbody>
</table>

**Dividing the locations into two groups**

The process of creating the fruit–location map for locations taken from sectors around the tree can be organized in the following algorithm.
**Algorithm 1.** Creation of the location combinations for given possible robot locations divided into $N_{loc}$ sectors.

**Input:** $N_{loc}$ sectors including the possible robot locations, bounding tolerance $t_b$

**Output:** total fruit–location map for all location combinations

1: The fruit–location map of sector 1 is taken

2: All discarded locations in the fruit–location map of sector 1 are found by the BB algorithm and eliminated

3: The remaining locations are taken as the total fruit–location map

4: **For** $i$ from 2 to $N_{loc}$

5: The total fruit–location map is combined with the fruit–location map of sector $i$

6: The location combinations are taken as the total fruit–location map

7: All discarded location combinations in the total fruit–location map are found by the BB algorithm and eliminated

This order of actions guarantees systematic creation and checking of the location combinations, adding the locations from the $i$th sector at each iteration. The resultant combinations are saved in the concurrent group. However, the size of the concurrent group grows rapidly with each iteration. For large values of $i$ the saving of combinations becomes significantly time- and memory-consuming.

An improvement which avoids the saving of a large array of combinations can be achieved by dividing the locations into two groups. According to this method, the optimal location combination search algorithm is changed to the following.
Algorithm 2. Creation of the location combinations for given possible robot locations divided into $N_{\text{loc}}$ sectors by dividing the locations into two groups.

**Input:** $N_{\text{loc}}$ sectors including possible robot locations, bounding tolerance $t_b$

**Output:** total fruit–location map for all location combinations

1: Fruit–location map 1 for locations in sectors $1 \ldots N_{\text{loc}}/2$ is found by Algorithm 1

2: Fruit–location map 2 for locations in sectors $N_{\text{loc}}/2+1 \ldots N_{\text{loc}}$ is found by Algorithm 1

3: Fruit–location map 1 is combined with fruit–location map 2 and taken as the total fruit–location map

4: All discarded location combinations in the total fruit–location map are found by the BB algorithm and eliminated

Algorithm 2 divides all of the locations into two groups (for example, with positive and negative X coordinate). Each group is divided into sectors and is treated separately, similar to the previous version of the algorithm. The advantage of this algorithm is that the resultant fruit–location maps are significantly smaller than the resultant fruit–location map in Algorithm 1.

In the following example, the robot location optimization for $N_{\text{loc}} = 8$ is performed for the RRP robot and CL apple tree model. The numbers of the considered location combinations derived from both algorithms are compared in Table 5.8. The number of location combinations is given for each iteration of the algorithms working on different sectors from 1 to 8. The values in the cells with (?) are unknown, since the computation took too much time and was stopped.

When running Algorithm 2, two fruit–location maps are created with 701 and 2521 location combinations, respectively. These maps are stored in arrays of sizes 144 x 701 and 144 x 2521, respectively. Then, to find the optimal location combination, the combinations from both maps are combined and checked by $O(701 \times 2521) = O(1767221)$ actions. This entire procedure is performed in 50 seconds.
Table 5.8 *Comparison of the fruit–location maps for Algorithm 1 and Algorithm 2.*

<table>
<thead>
<tr>
<th>Sector number</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Number of locations in the sector</strong></td>
<td>15</td>
<td>12</td>
<td>13</td>
<td>12</td>
<td>15</td>
<td>12</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td><strong>Number of locations in the sector after BB</strong></td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>10</td>
<td>6</td>
<td>6</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm 1</th>
<th>Considered sectors</th>
<th>1</th>
<th>1,2</th>
<th>1,2,3</th>
<th>1,2,3,4</th>
<th>1,2,3,4,5</th>
<th>1,2,3,4,5,6</th>
<th>1,2,3,4,5,6,7</th>
<th>1,2,3,4,5,6,7,8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Size of the fruit–location map</td>
<td>14</td>
<td>67</td>
<td>189</td>
<td>701</td>
<td>5294</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm 2</th>
<th>Considered sectors for fruit–location map 1</th>
<th>1</th>
<th>1,2</th>
<th>1,2,3</th>
<th>1,2,3,4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Size of fruit–location map 1</td>
<td>14</td>
<td>67</td>
<td>189</td>
<td>701</td>
</tr>
<tr>
<td></td>
<td>Considered sectors for fruit–location map 2</td>
<td>5</td>
<td>5,6</td>
<td>5,6,7</td>
<td>5,6,7,8</td>
</tr>
<tr>
<td></td>
<td>Size of fruit–location map 2</td>
<td>12</td>
<td>80</td>
<td>425</td>
<td>2521</td>
</tr>
</tbody>
</table>

When running Algorithm 1, a single fruit–location map is created. The numbers of location combinations produced by combining the first five sectors are 14, 67, 189, 701 and 5294. The combining of sector 6 took more than 1000 seconds, at which point the computation was stopped. This shows the inefficiency of Algorithm 1. Despite the fact that the number of location combinations achieved in Algorithm 1 (5294) was less than the total number achieved in Algorithm 2 (1767221), storage and other involved accesses to the computer memory took most of the computation time and made Algorithm 1 inefficient for the computation of a fruit–location map of more than approximately 5000 location combinations in size.
5.3.4. Improved optimal location combination search

The search algorithm, improved by all methods described in this chapter, formulated in Algorithm 3.

**Algorithm 3.** Optimization of the robot location.

**Input:** Tree model, robot kinematics, maximal number of the robot locations $N_{loc,max}$, bounding tolerance $t_b$, location grid resolution $r_{grid}$,

**Output:** optimal robot locations for $N_{loc} = 2 .. N_{loc,max}$

1: Create grid of possible robot locations with resolution $r_{grid}$

2: Create the fruit–location map

3: **For** $N_{loc} = 2 .. N_{loc,max}$

4: Apply Algorithm 2 with bounding tolerance $t_b$

5: Find location combination with minimal value of the cost function $F$ and set this combination as optimal robot locations for $N_{loc}$

An example of the results of Algorithm 3 is shown in Fig. 5.6. Each picture represents one iteration of the search algorithm. The locations in the considered sectors are shown as dots. Colored dots represent locations related to different sectors. Numbers without dots represent the discarded locations. The blue dots are related to sectors that have not yet been considered.
Fig. 5.6 Eight iterations of the algorithm for the optimal location combination search. The possible locations are represented by dots with different colors corresponding to the different sectors.

The method presented here uses analysis of the geometric features of the tree and algorithmic improvements to allow solving the robot location optimization for relatively complicated single trees. All robots optimized to perform harvesting on a single tree in Chapter 4 were found with the help of this method. However, the method becomes impractical for a large number of trees or an orchard—the typical case in agricultural robot optimization. This limitation requires the development of a method for environment simplification.
6. Development of the Characteristic Environment Method

To decrease the time needed to solve the optimization problem by methods presented in Sections 4 and 5 and applied on the original models of agricultural environment, we proposed a methodological simplification of the environment. Similar to the representation of a population by a sample in the statistical analysis, we used the assumption that an actual environment (AE) corresponding the population and consisting of a large number of targets can be replaced by an environment with a low number of targets characterizing the actual targets corresponding to the sample. This simplified artificial environment was defined as characteristic environment (CE) and the process of the CE building was defined as environment characterization.

The development of the method was based on the following assumptions.

1. Robot obstacles (e.g., branches) are not taken into consideration.
2. Small changes of the end effector location in the robot workspace cause small changes in the robot joint space. In general, this is true for all non-singular robot configurations.

The stages of the method development are presented in the following chart (Fig. 6.1).

![Diagram showing the stages of environment characterization method]

---

Fig. 6.1 Developmental stages of the environment characterization method.
Fig. 6.2 2D RR robotic arm for fruit picking (a). The robot is modeled by lines representing the robot links with the base at the origin (b). The robot reaches its targets, represented by red dots.

The method is developed in this chapter on a 2D environment with the robot presented in Fig. 6.2: a robot with 2 DOF approaches the fruit on a plane (Fig. 6.2a). The robot links are modeled by lines in Fig. 6.2b, and the fruit are modeled by dots.

6.1. Overview of characteristic fruit and cluster probability

In an AE, the fruit are concentrated in clusters representing areas with high fruit density, for example, fruit growing on a branch. If the cluster is small enough, and all of the fruit in it are located close together, according to assumption (2), the robot configurations for picking these fruit are also close (in the robot configuration space). For the 2D example shown in Fig. 6.3, the actual fruit in the small groups surrounded by ellipses have close robot configurations. Therefore, all robot configurations for picking the fruit from this cluster can be represented by a configuration that is typical to this cluster. The cost-function values of picking these fruit are also close and can be represented by the cost-function value of the characteristic robot configuration. Nevertheless, since the robot still has to be designed, the robot configuration cannot be used. Thus, the environment analysis can be made only in the workspace, and not in the robot configuration space.
Fig. 6.3 Small clusters of actual targets surrounded by red ellipses are characterized by the mean characteristic fruit, represented by green stars. The farthest fruit is represented by a black triangle.

The fruit that can be picked by the assumed characteristic robot configuration are the characteristic fruit (targets), representing the fruit cluster in the robot workspace. Since these fruit characterize the location of the cluster, their location is taken as the geometrical center of the cluster, and this characteristic fruit is called the mean characteristic fruit (target). In the 2D example shown in Fig. 6.3, they are represented by green stars.

Thus, the computation of the cost function for picking all fruit in the cluster is replaced by the computation of the cost function for the mean characteristic fruit. However, clusters containing different numbers of fruit influence the total cost function differently. The more fruit there are in the cluster, the more frequently the robot end effector has to visit this cluster, and to be fitted to work in this cluster. Thus, the clusters are also characterized by their fruit number, which is divided by the total fruit number for normalization. This characteristic is called cluster probability, $P_i$, i.e., the robot's probability of visiting cluster $i$.

However, while it provides a good evaluation of the cost function, the mean characteristic fruit can fail in characterizing the robot working volume dimensions. Simplification of the CE, which is one of the goals of the characterization, can be achieved by decreasing the number of characteristic fruit, thereby increasing the cluster size. But for some small number of characteristic fruit, the cluster size may become too big. Then, while approaching the
characteristic fruit in the center of the cluster, the robot might not be able to reach the extreme fruit of the cluster, which is needed to fulfil the optimization constraint.

To solve this problem and characterize the maximal robot reach, the extreme characteristic fruit (target) is used. If the robot base location is known, the extreme characteristic fruit is the fruit farthest in the robot workspace from the robot base location. However, since the robot base location is being sought during the optimization and, consequently, is unknown during the environment characterization, extreme characteristic fruit cannot be found, but have to be evaluated based on the robot workspace. In the 2D example shown in Fig. 6.3, the extreme fruit that is farthest from the robot base is represented by a black triangle.

To follow the guidelines formulated in Section 3.6.4 and avoid worst cases, the extreme fruit must not be taken as the fruit that are farthest from the robot base. If the robot has to approach no less than 95% of the fruit, it may not need to approach the remaining 5% of the fruit causing the highest values of the cost function. In this case, the extreme fruit must be the farthest fruit among the 95% of the nearest fruit.

Thus, according to the definition of the mean and extreme characteristic fruits, the mean characteristic fruit are responsible for minimization of the cost function, while the extreme characteristic fruit are responsible for satisfying the optimization constraint.

6.2. Description of the robot optimization problem for a 2D environment

In this section, we solve a simplified 2D robot optimization problem to illustrate the described method and to demonstrate the stages of its development.

6.2.1. Definition of the RR robot optimization problem

The 2D robotic arm with RR structure presented in Fig. 6.2 is optimized for fruit picking. The robot base is located in a single location at the origin of the coordinate system. The fruit are distributed on the XY plane. The number of actual fruit is $N_{act} = 12$, where fruit $i$ is located in the coordinate $(x_i,y_i)$ (Fig. 6.2a). The fruit mass, which represents the robot's load, is $m_i = 0.1$ kg.

The RR robot is a planar robot in the XY plane consisting of two links with lengths $l_1$ and $l_2$. The material of the links has a linear density $\rho = 1$ kg/m, such that the masses of the links are $m_i = \rho l_i$.
and $m_2 = \rho l_2$. The joint angles between the links are $\theta_1$ and $\theta_2$. To pick fruit $i$, the robot has to move its end effector from the home configuration to the point of the fruit's location $(x_i, y_i)$ (Fig. 6.2b). The joint angles needed to approach point $i$ are calculated by inverse kinematics ($\text{InvKin}$):

$$\text{InvKin}\begin{pmatrix} x_i \\ y_i \end{pmatrix} = \begin{pmatrix} \theta_{1,i} \\ \theta_{2,i} \end{pmatrix}, \quad \text{Eq. 6.1}$$

Gravity, with direction -Y, is applied on the masses of the links in the center of the links and the fruit. To simplify the problem, only the masses of the links and the load (fruit) are taken into consideration. The torques in the robot actuators are applied against the weight of the robot links, depending on the link lengths and joint angles. The torques in the actuators, $\tau_1$ and $\tau_2$, for the robot configuration with the joint angles $\theta_1$ and $\theta_2$ are

$$\tau_1 = g \left( \rho l_1 \frac{1}{2} \cos(\theta_1) + \rho l_2 \left( l_1 \cos(\theta_1) + \frac{l_2}{2} \cos(\theta_1 + \theta_2) \right) + m_i \left( l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) \right) \right), \quad \text{Eq. 6.2}$$

$$\tau_2 = g \left( \rho l_2 \frac{1}{2} \cos(\theta_1 + \theta_2) + m_i l_2 \cos(\theta_1 + \theta_2) \right)$$

where $g$ is the gravitational acceleration, and equals 10 N/kg in this research.

The robot's home configuration is the configuration from which it starts and to which it returns after each fruit picking. The home configuration is needed for placement of the picked fruit into the gathering bin. The home configuration $\Theta_{\text{home}}$ is taken as:

$$\Theta_{\text{home}} = \begin{pmatrix} \theta_{1,\text{home}} \\ \theta_{2,\text{home}} \end{pmatrix} = \text{InvKin}\begin{pmatrix} 1 \\ 0.5 \end{pmatrix}, \quad \text{Eq. 6.3}$$

The energy $E_i$ needed by the robot actuators to perform the motion between the home configuration, $\Theta_{\text{home}}$, and the configuration for picking the fruit $i$, $\Theta_i$, is

$$E_{i,j} = \int_{\Omega_i} \tau_1 d\Theta, \quad E_{x,i} = \int_{\Omega_i} \tau_2 d\Theta, \quad \text{Eq. 6.4}$$

where $\Omega_i$ is the robot trajectory from $\Theta_{\text{home}}$ to $\Theta_i$, which is linear in the robot configuration space.

The power of the robot actuators, $W = (W_1, W_2)$, is defined as follows: the power of the first actuator is $W_1 = 1$ W. The power of the second actuator is proportional to the mass that is actuated by it:
\[ W_2 = W_1 \frac{m_2 + m_l}{m_1 + m_2 + m_l} = W_1 \frac{\rho l_2 + m_l}{\rho l_1 + \rho l_2 + m_l} \]  
Eq. 6.5

The time needed to perform the motion from the home configuration to the configuration for fruit picking depends on the robot masses and lengths and the power of its actuators. The exact calculation of the time is complicated. To simplify the problem, the following approximation of motion time is used. The time needed for each actuator to perform the motion to fruit \(i\) is

\[ t_{1,i} = \frac{E_{1,i}}{W_i} \text{ and } t_{2,i} = \frac{E_{2,i}}{W_i}. \]

The total motion time is the maximum time needed for the actuators

\[ t_i = \max(t_{1,i}, t_{2,i}) \quad \text{Eq. 6.6} \]

The cost function of the 2D robot considered in this chapter, \(F_{2D}\), is the average fruit-picking time

\[ F_{2D} = \frac{\sum t_i(l_1, l_2, \theta_{1,i}, \theta_{2,i})}{N_{act}}, i = 1..N_{act} \quad \text{Eq. 6.7} \]

The optimization variables are the robot link lengths \(l_1\) and \(l_2\) and robot joint angles \(\theta_1\) and \(\theta_2\), which are constrained by the inverse kinematics.

The allowed percentage of unpicked fruit, \(a_{allow}\), defines the maximum amount of fruit that the robot can be unable to approach. The constraint of the optimization is the condition that the percentage of unpicked fruit \(a_{uf}\) be less than \(a_{allow}\).

The optimization problem has the following form:

\[ \min F_{2d}(l_1, l_2, \theta_{1,i}, \theta_{2,i}), i = 1..N_{act} \]

s.t. \( InvKin \left( \begin{array}{c} x_i \\ y_i \end{array} \right) = \left( \begin{array}{c} \theta_{1,i} \\ \theta_{2,i} \end{array} \right) \)

\[ a_{uf} \leq a_{allow} \quad \text{Eq. 6.8} \]

6.2.2. Analysis of the characterization

Since the goal of the environment characterization was simplification of the optimization process based on the AE, we wanted to know how many characteristic targets are needed to achieve the optimal robot based on a characteristic environment (RCE) that is close enough to the optimal robot based on the actual environment (RAE). Three proximity indices were defined to evaluate this: maximal target deviation, relative robot link difference and relative robot cost function.
Maximal target deviation $dP$ was defined as the maximal distance between a characteristic target and the characterized actual targets:

$$dP = \max_{i} \left( \max_{j} \left( \sqrt{(x_{i,j,\text{act}} - x_{i,\text{char}})^2 + (y_{i,j,\text{act}} - y_{i,\text{char}})^2} \right) \right)$$  \hspace{2cm} \text{Eq. 6.9}

where the actual targets in the cluster $i$ are characterized by characteristic target $i$, and $j$ runs from 1 to $C_i$, the number of actual targets in cluster $i$.

The relative robot link difference $dL$ was defined as follows:

$$dL = \sqrt{\left( l_{1,\text{act}} - l_{1,\text{char}} \right)^2 + \left( l_{2,\text{act}} - l_{2,\text{char}} \right)^2} \quad \frac{l_{1,\text{act}} + l_{2,\text{act}}}{l_{1,\text{act}} + l_{2,\text{act}}}$$  \hspace{2cm} \text{Eq. 6.10}

where $l_{1,\text{act}}$ and $l_{2,\text{act}}$ are the link lengths of the RAE, and $l_{1,\text{char}}$ and $l_{2,\text{char}}$ are the link lengths of a RCE.

Relative robot cost function difference $dF$ was defined as follows:

$$dF = \left| \frac{F_{2D}(l_{1,\text{act}}, l_{2,\text{act}}) - F_{2D}(l_{1,\text{char}}, l_{2,\text{char}})}{F_{2D}(l_{1,\text{act}}, l_{2,\text{act}})} \right|$$  \hspace{2cm} \text{Eq. 6.11}

where $F_{2D}(l_{1,\text{act}}, l_{2,\text{act}})$ is the RAE cost function, and $F_{2D}(l_{1,\text{char}}, l_{2,\text{char}})$ is the RCE cost function.

6.3. Environment characterization

6.3.1. Mean target characterization

The targets comprising the environment were divided into clusters by K-mean clustering method (Press et al., 2007). The targets characterizing the clusters were located in the mass center of the clusters. These characteristic targets are called mean characteristic targets, and their number is depicted as $N_{\text{mean}}$. Clustering into 1, 2, 3 and 4 clusters with mean characteristic targets in their mass centers is illustrated in Fig. 6.4. The percentage shown for each characteristic target represents the cluster probability $P_i$. The maximal target distances between the mean characteristic targets and the characterized actual targets are 0.62, 0.41, 0.36 and 0.28 for 1, 2, 3 and 4 clusters, respectively.
Fig. 6.4 *Clustering of an environment. The points represent the actual targets, the asterisks represent the mean characteristic targets. The percentage of actual targets characterized by each mean characteristic target appears beside the asterisk.*

The optimal RAE and RCE for the number of clusters $N_{\text{mean}} = 1 \ldots 12$ were found and compared. The dependence of $dP$, $dF$ and $dL$ on $N_{\text{mean}}$ is shown in Fig. 6.5. The percentage of fruit that were not picked by the RCE performing in the AE, $\alpha_{uf}$, also depends on $N_{\text{mean}}$. Increasing $N_{\text{mean}}$ decreases all of the proximity indices between the RAE and RCE, as well as $\alpha_{uf}$ of the RCE performing in the AE. Starting from $N_{\text{mean}} = 7$, the RCE is identical to the RAE.

Target approaches were compared for RAE and RCE (Fig. 6.6). The RCE was based on a CE consisting of 1, 3 and 6 mean characteristic targets. While the RCE based on a CE with $N_{\text{mean}} = 1$ was far from the RAE, the difference between the RAE and the RCE based on a CE with $N_{\text{mean}} = 3$ was acceptably small. The difference for $N_{\text{mean}} = 6$ was even smaller, but such a large $N_{\text{mean}}$ may not justify the computational effort. Hence, there was an optimal value of $N_{\text{mean}}$ providing a RCE acceptably close to the RAE while keeping the amount of computations relatively small.

Fig. 6.5 *The dependence of $dP$, $dF$, $dL$ and $\alpha_{uf}$ on $N_{\text{mean}}*. 
Fig. 6.6 *Optimal robots based on the actual environment (RAE) (solid lines) and on the characteristic environment (RCE) (dashed lines) achieved for characteristic environments (CE) with different numbers of clusters: 1, 3 and 6. The actual targets constituting the actual environment (AE) are represented as dots, the characteristic targets constituting CE are represented as asterisks. The home configuration is shown by solid continuous lines for RAE and dashed lines for RCE.*

The analytical dependencies of $dP$, $dL$ and $dF$ on $N_{mean}$ could not be obtained because of the complexity of the expressions describing the inverse kinematics. Hence, these dependencies were evaluated numerically. For this purpose, 100 random AE were simulated, each with 12 actual targets whose locations were uniformly distributed in the domain $1 \leq X \leq 2, 1 \leq Y \leq 2$. As described above, the RAE for each AE was found and compared with the RCE based on the CE with $N_{mean} = 1...12$. An analysis for the characterization in this example is presented in Appendix B.1. The results showed that increasing the number of characteristic targets decreases the difference between the RAE and RCE, meaning that a CE with more characteristic targets yields a better characterization of the AE. This result was expected, since additional characteristic targets provide additional information about the spatial distribution of the actual targets, leading to a more precise RCE.

### 6.3.2. Extreme target characterization

Testing the performance of a RCE based on a CE with a low $N_{mean}$, we saw that the RCE achieves a cost-function value close to the RAE, but the RCE can miss targets. For example, we
take a RCE achieved by a CE with $N_{\text{mean}} = 2$. The RCE shown in Fig. 6.7a has a low cost-function difference ($dF = 4.2\%$). However, the RCE is unable to reach two targets—$\alpha_{uf} = 16\%$—meaning that it does not fulfill the optimization constraint $\alpha_{uf} < \alpha_{\text{allow}}$. This RCE can reach the mean characteristic targets in the cluster centers, but its total length is insufficient to reach the farthest fruit marked by circles in Fig. 6.7a.

If the robot is located close to the targets, as in Fig. 6.7b, $dF = 10\%$, but $\alpha_{uf} = 41\%$, and the RCE cannot reach the nearest and farthest targets.

To characterize the limits of the robot workspace, the farthest targets can be added to the CE. The nearest targets are also used for the characterization since, if the robot is located close to the targets, the close targets can also cause unsolvable inverse kinematics. In these examples, the farthest and nearest targets represent the worst cases—the targets that are hardest to reach. Since the robot and the environment are relatively simple, we assume that the farthest and nearest targets characterize the limits in the AE and are the extreme characteristic targets. The more general case will be further discussed in Section 6.4.3.

Fig. 6.7 Comparison of the performances of optimal robots based on the actual environment (RAE) and on the characteristic environment (RCE) in the actual environment. The robot base is located far (a) and close (b) to the targets. The targets that are not approached by the RCE are marked by black circles.
Using the extreme characteristic targets in a CE does not contradict guideline 1 formulated in section 3.6.4. If \( \alpha_{\text{allow}} = 0 \), all of the targets, including the worst cases, must be approached. If \( \alpha_{\text{allow}} > 0 \), the extreme characteristic targets are not the extreme targets of the environment, but the extreme targets in a subset of actual targets that is sufficiently large to satisfy the optimization constraint (namely, \( \alpha_{\text{uf}} < \alpha_{\text{allow}} \)). This will be discussed in further detail in Section 6.4.1. The number of extreme characteristic targets is depicted as \( N_{\text{ext}} \), and consists of the numbers of the farthest extreme targets, \( N_{\text{far}} \), and nearest extreme targets \( N_{\text{near}} \): \( N_{\text{ext}} = N_{\text{far}} + N_{\text{near}} \).

The farthest and nearest targets cannot be reached by a robot because of its limited reach and the proximity of its base to the targets, respectively. These two cases can be detected by the distance of the targets from the robot base in the Cartesian workspace. The targets that are farthest from and nearest the robot base from the previous example are shown in Fig. 6.8. The limits of the robot's reach are marked by two arcs whose center is the robot base. The unreachable targets are located outside the area delimited by the arcs. Consequently, the robot's location must be known to characterize the extreme targets in the AE.

Optimal robots based on a CE including only the extreme targets are shown in Fig. 6.9. The AE was characterized by two combinations: \( N_{\text{far}} = 1, N_{\text{near}} = 0 \) in (a) and \( N_{\text{far}} = 1, N_{\text{near}} = 1 \) in (b). The proximity indices between the RAE and RCE are shown in Table 6.1.

Fig. 6.8 The targets that are unreachable by the optimal robot based on the characteristic environment (RCE) in Fig. 6.7 are outside the limits defined by the arcs with the robot base at their center as they pass through the extreme targets.
Fig. 6.9 Optimal robots based on the actual environment (RAE) and on a characteristic environment (RCE) with extreme characteristic targets: $N_{\text{far}} = 1$, $N_{\text{near}} = 0$ in (a), and $N_{\text{far}} = 1$, $N_{\text{near}} = 1$ in (b). The actual targets are represented as dots, the farthest extreme characteristic targets are represented as $\Delta$, the nearest extreme characteristic targets are represented as $\triangledown$.

The analysis for the characterization in this example is presented in Appendix B.1B.2. According to the results, the RCE based on the extreme targets can reach all targets of the AE fulfilling the optimization constraint $\alpha_{uf} < \alpha_{allow}$. However, the difference in the cost function, $dF$, is larger for the RCE based on the extreme characteristic targets than for the RCE based on the mean characteristic targets.

Table 6.1 Comparison of the optimal robots based on the actual environment (RAE) and on a characteristic environment (RCE) with extreme characteristic targets.

<table>
<thead>
<tr>
<th></th>
<th>$N_{\text{far}}$</th>
<th>$N_{\text{near}}$</th>
<th>$dF$ (%)</th>
<th>$dL$ (%)</th>
<th>$\alpha_{uf}$ (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>19.6</td>
<td>2.5</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>4.4</td>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>
According to the definition of the two types of characteristic targets (mean and extreme), the mean characteristic targets are used to fit the value of the cost function of a RCE to the RAE, while the extreme characteristic targets are used to fulfill the constraint of the optimization. Considering the results for the simple case of the RR robot optimization, we can say that the extreme characteristic targets define the total length of the RR robot links, while the mean characteristic targets define the ratio between the link lengths.

### 6.3.3. Combined characterization

The advantages of the characterization based on mean and extreme characteristic targets can be realized by combining both types of characterization. CE including both mean and extreme targets are presented in Fig. 6.10.

![Fig. 6.10 Optimal robots based on the actual environment (RAE) and on a characteristic environment (RCE) with different numbers of mean characteristic targets, \(N_{\text{mean}} = 1\ldots4\), and extreme characteristic targets with \(N_{\text{far}} = 1\), \(N_{\text{near}} = 1\). The actual targets are represented as dots, the mean characteristic targets as asterisks, the farthest extreme characteristic targets as \(\Delta\), the nearest extreme characteristic targets as \(\\slash\).](attachment:image.png)
Table 6.2 Comparison of the optimal robots based on the actual environment (RAE) and on a characteristic environment (RCE) with mean and extreme characteristic targets.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>(N_{\text{mean}})</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>(N_{\text{far}})</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(N_{\text{near}})</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(\text{dF (%)})</td>
<td>4.3</td>
<td>2</td>
<td>2</td>
<td>1.3</td>
</tr>
<tr>
<td>(\text{dL (%)})</td>
<td>1.7</td>
<td>3.9</td>
<td>3.9</td>
<td>2.5</td>
</tr>
<tr>
<td>(\alpha_{\text{auf (%)}}</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The proximity indices between the RAE and RCE are shown in Table 6.2. For \(N_{\text{mean}} > 2\), the combined characterization is better than the characterization using only the mean characteristic targets with the same number of characteristic targets shown in Fig. 6.5.

The analysis for the characterization in this example is presented in Appendix B. According to the results, the combined characterization ensures fulfillment of the constraint (in the simple case with the RR robot) and provides a low \(dF\).

6.4. Environment characterization with allowed unpicked fruit

6.4.1. Dependence of the robot cost function on \(\alpha_{\text{allow}}\)

Starting from this section, we assume that some number of targets is allowed to be unreachable, \(\alpha_{\text{allow}} > 0\). The number of allowed unreachable fruit is \(N_{\text{allow}} = N_{\text{act}} \cdot \alpha_{\text{allow}}\). To minimize the cost function in the case of \(\alpha_{\text{allow}} > 0\), the robot has to omit \(N_{\text{allow}}\) targets with the largest cost-function values. Accordingly, the robot performance cost-function \(F\) for \(N_{\text{allow}} > 0\) is calculated as follows: if a robot can reach more than \(N_{\text{act}} \cdot N_{\text{allow}}\) targets, it reaches only the \(N_{\text{act}} \cdot N_{\text{allow}}\) targets with the lowest cost-function values, which are averaged to calculate the cost-function for the entire environment.

An example of optimal robots for \(\alpha_{\text{allow}} = 0\), \(\alpha_{\text{allow}} = 10\%\), and \(\alpha_{\text{allow}} = 20\%\) is presented in Fig. 6.11, where the number of actual fruit \(N_{\text{act}} = 30\). The cost-function values for these robots are 9.5, 8.8, and 8.2, respectively. The more fruit the robot is allowed not to reach, the shorter and faster it is.
Fig. 6.11 Optimal robots for $\alpha_{\text{allow}} = [0, 10\%, 20\%]$. The unpicked fruit are marked by circles.

To evaluate the sensitivity of the robot performance cost function $F_{2D}$ to the percentage of allowed unreachable fruit $\alpha_{\text{allow}}$, the dependence $F_{2D}(\alpha_{\text{allow}})$ was checked for 100 random environments with $N_{\text{act}} = 30$. The results are presented in Fig. 6.12.

Fig. 6.12 Dependence of the robot performance cost function $F_{2D}$ on the allowed percentage of unpicked fruit, $\alpha_{\text{allow}}$, for 100 random actual environments (AE). The thin lines represent the dependence for a specific AE. The thick red line represents the average.
The graphs of the dependencies decrease monotonically, representing the trade-off between the robot's average time performance (cost function) and the percentage of unpicked fruit. This trade-off means that if a general economic cost function is defined to include the robot performance time and the unpicked fruit percentage, there is an optimal percentage of unpicked fruit $a_{allow}$. In this work, we use $a_{allow}$ in the interval $[0\ldots15\%]$ depending on the considered case.

### 6.4.2. Environment characterization with allowed unreachable targets

Both mean and extreme characteristic targets can be affected by the allowed percentage of unpicked fruit, $a_{allow}$. Nevertheless, choosing extreme targets that are not included in the allowed unreachable targets is critical. The simplest way to choose the extreme targets is as follows: exclude the farthest and nearest targets from the actual targets being considered, and choose the extreme targets as the farthest and nearest among the remaining actual targets.

In the following example, we took the number of actual targets $N_{act} = 30$ and set the percentage of allowed unpicked targets $a_{allow} = 10\%$; to fulfill the constraint, the optimized robot may not reach $N_{allow} = \lfloor N_{act} \cdot a_{allow} \rfloor = \lfloor 30 \cdot 0.1 \rfloor = 3$ targets, where $N_{allow}$ is the number of allowed unreachable targets. Then the farthest extreme target is the fourth farthest target as shown in Fig. 6.13.

![Environment characterization with allowed unreachable targets](image)

*Fig. 6.13 Environment characterization with allowed unreachable targets. The farthest unreachable targets are marked by squares. The farthest characteristic target is marked by Δ. The actual reachable targets are bounded by an arc with the center in the robot base location passing through the farthest characteristic target.*
6.4.3. Detecting the extreme targets with allowed unreachable targets

In the example of the characterization by extreme targets in Section 6.3.2 we assumed that the farther or nearer the target is from the robot base, the higher the robot performance cost function $F_{2D}$. Nevertheless, in the general case, the cost function $F_{2D}$ depends non-monotonically on the distance of the target from the robot base location. An example of a mapping between the location of a target and the cost function for reaching that target is shown in Fig. 6.14. For this example, we take a robot with the kinematics, base location and home configuration shown in Fig. 6.14a. For each target location $(x,y)$, the cost function $F(x,y)$ (the time needed to move from the home configuration to the reaching configuration for point $(x,y)$) is calculated. The surface of the function $F(x,y)$ on the domain $0 \leq x \leq 2$, $0 \leq y \leq 2$ is shown in Fig. 6.14b. The contours of this surface are shown in Fig. 6.14c. To illustrate the relationship between $F$ and the distance of the target from the robot base, the circles with the centers in the robot base are also presented in Fig. 6.14c. In the area distant from the robot base, $F_{2D}$ increases monotonically with the target distance (similar to the assumption for the example in Section 6.3.2). In the area close to the robot base, the contours of $F_{2D}$ intersect the circles, meaning that there is no obvious dependence between $F_{2D}$ and the target distance.

Fig. 6.14 Mapping between the target location and the robot cost function for reaching this target. A robot with a specific kinematics and base location is taken and shown in (a). The surface of the cost function depending on the target location $F_{2D}(x,y)$ in the domain $0 \leq x \leq 2$, $0 \leq y \leq 2$ is shown in (b). The contours of the surface are compared with the concentric circles with the robot base as center in (c).
Since the extreme characteristic targets define the extreme robot configurations and the unreachable targets, we could conclude from this example that finding the extreme targets based only on the distance of the target from the robot base might not be effective. This is especially true for cases with more complicated robot kinematics. Nevertheless, the robot kinematics and consequently, its cost function mapping, are unknown during the designing. Hence, only the features of the workspace can be used for the environment characterization.

To increase the effectiveness of the extreme targets' characterization, we used the following improvement of the method based on the distance from the robot base location. According to the mapping in Fig. 6.14, targets that are equally distanced from the robot base but located in different areas of the workspace can have different cost-function values. These values and the extreme target locations are different for different robots and cannot be known during the optimization. Therefore, we check the extreme targets in different areas of the workspace, assuming that one of them is the true extreme target for the optimal robot.

According to assumption 2 (at the beginning of Chapter 6), targets that are close to each other in the robot workspace are close in the robot joint space causing similar cost-function values. Hence, to find targets with different cost-function values, the areas of the workspace where these targets are located must be as distant from each other as possible. This requirement resembles the definition of clustering: find areas as close to their centers as possible. We thus use clustering to find the targets that are most distant from each other, taking them as the extreme target of each cluster. Extreme characterization according to this method can be combined in a natural way with the mean characterization.

### 6.5. Environment characterization using mean and extreme targets

An example of characterization by a number of extreme targets combined with mean targets is shown in Fig. 6.15. The environment with $\alpha_{allow} = 10\%$ is taken from the previous example. The mean and extreme targets are found using clustering into two (Fig. 6.15a) and three (Fig. 6.15b) clusters. The environment is divided into clusters marked by different colors. The asterisks in the cluster centers represent the mean characteristic targets. The number of allowed unreachable targets $N_{allow} = 3$. In each cluster, the three farthest targets are considered as unpicked, and thus the fourth farthest target is taken as the extreme farthest characteristic target. In the same way,
the three nearest targets in each cluster are considered as unpicked, and thus the fourth nearest target is taken as the extreme nearest characteristic target. Thus, four and six farthest and nearest targets are achieved by the clustering.

According to our assumption, one of the farthest and nearest targets represent the true extreme targets causing the extreme values of the cost function. Hence, to check for the true extreme targets during robot optimization, we have to check all of the extreme characteristic targets. The more farthest and nearest targets that are checked during the optimization, the closer one of them will be to the true extreme target.

The RAE reaching the targets is shown in Fig. 6.15c. The three unreached targets are marked by black circles. In both cases of clustering, the unreached targets are indeed the farthest targets located in one of the clusters.

The number of clusters defining the number of characteristic targets depends on the required proximity of the RCE to the RAE. Choice of cluster number is investigated for more complicated AE in Section 6.7.

![Fig. 6.15](image)

*Fig. 6.15* Characterization of an environment with \( \alpha_{\text{allow}} = 10\% \) by mean targets and extreme targets achieved by clustering into two (a) and three (b) clusters. The optimal robot based on the actual environment reaching the targets is shown in (c), and the unreached targets are marked by circles.
The further development of the environment characterization is based on using the mean and extreme targets, such that their number will be equal \( N_{\text{mean}} = N_{\text{far}} = N_{\text{near}} \) and equal to the number of clusters. Hence, the number of clusters will be depicted as \( N_{\text{clust}} \).

6.6. Metrics for calculating the extreme characteristic targets

Since the extreme targets could not be calculated in the robot joint space, we approximated them by the extreme targets in the workspace. This approximation was successful for the considered simple case, when the targets could be reached from any direction. However, if the reaching direction is important, for example, if the robot end effector has to be perpendicular to the fruit, the extreme robot configuration depends on the fruits’ orientation.

An example of a strong influence of fruit orientation on the robot kinematics is shown in Fig. 6.16. A robot with a RRR structure picks two fruit on a plane with different location and orientation. Fruit 1 is oriented perpendicular to the line connecting it with the robot base, and fruit 2 is oriented along the line connecting it with the robot base. To pick fruit 1 and 2, the robot has to be in configurations 1 and 2, respectively. In the case presented in Fig. 6.16a, fruit 1 is located furthest from the robot base. Nevertheless, configuration 2 represents an extreme configuration of the robot, because the robot moves a longer trajectory and has higher actuator torques in the picking configuration. In the same way, in the case presented in Fig. 6.16b, configuration 1 represents an extreme configuration of the robot despite the fact that fruit 2 is the fruit nearest to the robot base.

![Fig. 6.16](image)

**Fig. 6.16** Extreme farthest (a) and nearest (b) fruit with similar locations but different orientations define different robot configurations. \( P_{\text{target}} \), \( V_{\text{target}} \) and \( P_{\text{base}} \) are the target location, target orientation vector and robot base location, respectively.
Hence, to find the extreme fruit characterizing the extreme robot configurations, the metrics for calculating the farthest and nearest fruit must be expanded to include the orientation of the fruit. The target orientation for the farthest targets is measured relative to the line connecting the robot base and the target. This line coincides with the totally stretched out robot with minimal required reach. This robot configuration is possible only when the target is perpendicular to this line. Thus, the distance in the extended metrics is larger when the deviation of the target orientation from the perpendicular position is larger. The deviation of the target orientation is calculated as follows:

\[
dV = \text{acos} \left( \frac{V_{\text{target}} \cdot (P_{\text{target}} - P_{\text{base}})}{||V_{\text{target}}|| \cdot ||P_{\text{target}} - P_{\text{base}}||} \right)
\]

Eq. 6.12

where \( P_{\text{target}} \), \( V_{\text{target}} \) and \( P_{\text{base}} \) are the target location, target orientation vector and robot base location, respectively. The Cartesian distance between the robot and the target is \( dR = ||P_{\text{target}} - P_{\text{base}}|| \). Then the distance for the farthest targets in the extended metrics is \( d = dR + w \cdot dV \), and the distance to the nearest targets is \( d = dR - w \cdot dV \), where \( w \) is the weight of the deviation of the target orientation. The weight \( w \) connects the length in meters and angle in radians. The contribution of the angle deviation particularly depends on the length of the last robot link responsible for the approaching orientation: the shorter the last link is, the lower the contribution of angle deviation.

### 6.7. Characterization of AE

In this section, the method is applied to more complicated robot-optimization problems, consisting of 3D models of actual plants having different structures and numbers of targets, and 3-DOF robots with different kinematics. The optimization is performed on an environment without obstacles.

To define the number of clusters for each given AE, we use the maximal cluster size, depicted as \( D_{\text{clust}} \). This parameter limits the maximal distance between two targets in a cluster. The value of \( D_{\text{clust}} \) is investigated in this section. The process of environment characterization is defined as the following Algorithm 4.
Algorithm 4. Environment characterization by clustering and extreme target.

**Input:** Given $\text{AE}$, $\alpha_{\text{allow}}$, $D_{\text{clust}}$, and $P_{\text{base}}$

**Output:** CE including mean and extreme targets

1: Calculate $N_{\text{clust}}$ according to $D_{\text{clust}}$
2: Divide the targets of the AE into $N_{\text{clust}}$ clusters
3: Define the characteristic mean targets of the clusters
4: Calculate $N_{\text{allow}}$ according to $\alpha_{\text{allow}}$
5: For each cluster
6: Correct the cluster by the metrics including target orientation
7: Define the characteristic extreme targets as the $N_{\text{allow}}+1^{\text{th}}$ farthest and nearest targets

The proximity between the RAE and RCE is evaluated by the proximity indices $dF$ and $dL$ defined in Section 6.2.2. The optimization solution runtimes for finding the RAE and RCE are also compared. The ratio between the time needed to find RAE and that needed to find RCE is depicted as TR.

**Example on TS apple tree model**

For the first example, we used the most simple tree model in the tree model library described in Appendix A: a TS apple tree with 30 fruit. Views of the tree model are presented in Fig. 6.17.
Fig. 6.17 Views of the Tall Spindle apple tree model: with obstacles (a), without obstacles (b). The fruit, presented as lines, are shown in side (c) and front (d) views.

RAE

The RAE with different robot types are shown in Fig. 6.18 while reaching the targets from optimal locations. The following robot structures were considered in this example: RRR (a), RRP (b) and PPP (c). The number of robot locations around the tree was $N_{loc} = 2$.

Fig. 6.18 Optimal robot based on the actual environment for Tall Spindle apple tree model with different structures: RRR (a), RRP (b) and PPP (c).
Environment characterization

Three CEs were built for the following maximal cluster sizes: $D_{\text{max,clust}} = 1 \text{ m}$, $D_{\text{max,clust}} = 0.4 \text{ m}$ and $D_{\text{max,clust}} = 0.2 \text{ m}$, corresponding to 2, 4 and 6 clusters, respectively. The CE are shown in Fig. 6.19, Fig. 6.20 and Fig. 6.21 in the frontal, side and isometric views. The actual fruit are represented as dots. The dots with different colors are related to different clusters. The mean characteristic targets located in the center of the clusters are represented as asterisks. The percentages near the asterisks show which part of the total number of actual fruit, $N_{\text{act}}$, is included in the cluster.

In these examples, we assumed some robot base location $P_{\text{base}}$. The extreme characteristic targets of the clusters were calculated relative to this robot base location. The extreme farthest and nearest targets of the clusters are represented as $\Delta$ and $V$, respectively. In a number of cases, when the number of actual fruit constituting a cluster was low, the farthest fruit in the cluster was coincident with the nearest fruit in this cluster. Since only one of them can represent the actual extreme fruit, this coincidence does not contradict the characterization of the extreme targets.

**Fig. 6.19** Characteristic environments for the Tall Spindle apple tree’s actual environment with 2 clusters. The actual fruit are depicted by dots with different colors corresponding to different clusters. The mean targets of the clusters are depicted as asterisks with the cluster percentage. The farthest and nearest extreme fruit of the clusters are depicted as $\Delta$ and $V$, respectively. The assumed robot base position is depicted by a circle.
Fig. 6.20  Characteristic environments for the Tall Spindle apple tree's actual environment with 3 clusters. All signs are as in Fig. 6.19.

Two additional examples of the tree models representing AE and their CE for the YT apple tree model including 67 fruit and the CL apple tree model including 144 fruit are described in Appendix B.4.

Fig. 6.21  Characteristic environments for the Tall Spindle apple tree's actual environment with 6 clusters. All signs are as in Fig. 6.19.
6.8. Comparison of RAE and RCE

The performances of RAE and RCE in the AE are compared in Table 6.3 by the proximity indices and the optimization runtime ratio. The RCE with $dF > 10\%$ are shaded in gray. For all combinations of environment and robot structure type, except for the PPP on YT, there exists a number of clusters that provide a sufficiently good characterization with $dF < 10\%$. The optimization based on the CE is solved at least four times faster for the TS model, 19 times faster for the YT model, and 40 times faster for the CL model than that based on the AE.

There are cases in which the difference $dF$ does not decrease when the number of clusters increases. This can be explained by the high non-linearity of the robot performance cost function and the complexity of the environment, in contrast to the simplified examples given elsewhere in Chapter 6. Nevertheless, in general, the main principle of the environment characterization remains correct: the more information used for the characterization, the more accurate the results achieved by the characterization.

If we assume that a difference in the cost-function value of $dF < 10\%$ is sufficiently good for the characterization, we can infer the maximal cluster size $D_{max,clust}$ inducing such values of $dF$ for different environments. Considering the shaded cells, we achieve the following values: $D_{max,clust} = 0.2$ for the TS tree, $D_{max,clust} = 0.2$ for the YT tree, and $D_{max,clust} = 0.5$ for the CL tree.

6.9. Optimization based on the worst case

In previous studies, other methods for environment simplification were used. One of them was to find and use the worst cases. A robot optimized for these cases is able to perform the task in the rest of the cases. For example, a robot approaching the most inconvenient fruit can also approach the rest of the fruit. This environment characterization method is similar to the characterization by the extreme characteristic fruit.
Table 6.3 *Comparison of the optimal robots based on the actual environment and those based on a characteristic environment with different numbers of clusters by the proximity indices (shaded cells with $dF>10\%$).*

<table>
<thead>
<tr>
<th></th>
<th>PPP</th>
<th>RRP</th>
<th>RRR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$N_{clust}$</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>$D_{max,clust}$ (m)</td>
<td>1</td>
<td>0.4</td>
</tr>
<tr>
<td></td>
<td>$dF$ (%)</td>
<td>11</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>$dL$ (%)</td>
<td>24</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>$\alpha_{uf}$ (%)</td>
<td>3.3</td>
<td>3.3</td>
</tr>
<tr>
<td></td>
<td>TR</td>
<td>19</td>
<td>13</td>
</tr>
<tr>
<td>TS, $N_{loc}=2$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$N_{clust}$</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>$D_{max,clust}$ (m)</td>
<td>0.5</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>$dF$ (%)</td>
<td>17</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>$dL$ (%)</td>
<td>16</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>$\alpha_{uf}$ (%)</td>
<td>7.4</td>
<td>4.4</td>
</tr>
<tr>
<td></td>
<td>TR</td>
<td>92</td>
<td>78</td>
</tr>
<tr>
<td>YT, $N_{loc}=4$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$N_{clust}$</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>$D_{max,clust}$ (m)</td>
<td>1</td>
<td>0.75</td>
</tr>
<tr>
<td></td>
<td>$dF$ (%)</td>
<td>16</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>$dL$ (%)</td>
<td>19</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>$\alpha_{uf}$ (%)</td>
<td>15.2</td>
<td>4.8</td>
</tr>
<tr>
<td></td>
<td>TR</td>
<td>253</td>
<td>161</td>
</tr>
<tr>
<td>CL, $N_{loc}=4$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$N_{clust}$</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>$D_{max,clust}$ (m)</td>
<td>1</td>
<td>0.75</td>
</tr>
<tr>
<td></td>
<td>$dF$ (%)</td>
<td>17</td>
<td>1.4</td>
</tr>
<tr>
<td></td>
<td>$dL$ (%)</td>
<td>15</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>$\alpha_{uf}$ (%)</td>
<td>8.3</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>TR</td>
<td>77</td>
<td>47</td>
</tr>
</tbody>
</table>

The results of the robot optimization based on the worst case were compared with the results of the optimization based on the characteristic task. The definition of “worst case” depends on the requirements of the robot designer. For example, we took the characteristic environment with...
only extreme characteristic fruit for $\alpha_{allow} = 5\%$ as an environment consisting of the worst cases. The difference between the RAE, RCE and a robot based on the environment consisting of the worst cases (RWCE) is presented in Table 6.4 for the CL apple tree model.

For all combinations of robot structure and number of clusters, the difference between the RAE and RCE was smaller than the difference between the RAE and RWCE (several fold smaller in a number of cases). This example demonstrates the limitation of using the worst cases for the environment simplification.

Table 6.4 *Relative difference, dF, between the performance of the optimal robots based on the actual environment and characteristic environment (RAE and RCE, respectively), and RAE and a robot based on the environment consisting of the worst cases (RWCE) in an actual environment for the Central Leader apple tree model.*

<table>
<thead>
<tr>
<th>dF (%)</th>
<th>$N_{clus}$</th>
<th>PPP</th>
<th>RRP</th>
<th>RRR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Nloc = 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RCE</td>
<td>21</td>
<td>2.2</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>RWCE</td>
<td>54</td>
<td>18</td>
<td>7.9</td>
</tr>
<tr>
<td></td>
<td>Nloc = 4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RCE</td>
<td>16</td>
<td>2.5</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>RWCE</td>
<td>35</td>
<td>4</td>
<td>7.6</td>
</tr>
<tr>
<td></td>
<td>Nloc = 6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RCE</td>
<td>16</td>
<td>4.3</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>RWCE</td>
<td>25</td>
<td>19</td>
<td>56</td>
</tr>
<tr>
<td></td>
<td>Nloc = 8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RCE</td>
<td>4.3</td>
<td>14</td>
<td>8.3</td>
</tr>
<tr>
<td></td>
<td>RWCE</td>
<td>30</td>
<td>29</td>
<td>50</td>
</tr>
</tbody>
</table>

104
7. Reaching Cones

The reaching cones characterize the obstacle-free space where the robot end effector can reach its target. The reaching cones are used to avoid the need to solve the robot motion planning problem, speed up the solution of the robot optimal location problem using reachability analysis, and speed up the solution of the robot kinematics optimization problem using environment characterization.

Circular cones, as shown in Fig. 7.1, were used to simplify the description of the reaching space. The circular cones were chosen as the simplest geometrical bodies fitting this description. They are defined by the vertex located in the fruit, vector $\vec{n}$ of the direction of the cone axis, the opening angle $\alpha$ and the cone height $h$. The cone parameters are derived from three points of intersection between the cone surface and the obstacles.

Fig. 7.1 Parameters of a circular reaching cone. The cone vertex is located at the fruit marked in yellow, the cone axis is defined by the vector $\vec{n}$, the cone opening angle is $\alpha$, the cone height is $h$, the surface is defined by three points of intersection with the obstacles depicted by the asterisks.
7.1. Development of the star map method for the reaching cone search

The reaching cones of all fruit on a tree can be calculated by checking all combinations of three branches for each fruit. The total number of operations needed to calculate the cones for a tree model is \( (N_{fruit}N_{branch}^3) \). For the considered tree models, which include hundreds of fruit and branches, this calculation can take an unacceptable amount of time, and thus it is not used. Therefore, a faster method is developed in this chapter.

An approximated search method based on the discretization of the search space with a particular resolution was used to find the reaching cones. The search space resembled a star map seen from some point, hence, the method was called the star map reaching cone search.

The reaching cone consists of rays representing straight obstacle-free trajectories of the robot end effector. These rays start from the fruit and grow in directions defined by pairs of the azimuth and polar angles \((\theta, \phi)\) with the origin in the fruit, as in Fig. 7.2. If a ray defined by a pair \((\theta, \phi)\) can grow to infinity, it can be included in one of the fruit reaching cones. If the ray reaches an obstacle, it cannot be included in a reaching cone. This condition was used as a rule to create the obstacle map (star map) for each fruit: for all combinations of ray angles \((\theta, \phi)\) taken with a given resolution, find whether the ray reaches an obstacle. For each ray reaching an obstacle, the distance to the obstacle is noted.

If a ray is tangent to an obstacle, representing the boundary between the obstacle-free space and the obstacle, the ray lies on the surface of the reaching cone. Once the star map is calculated, three boundary rays representing three points of intersection between the cone and the obstacles can be found to define a reaching cone.

![Ray Diagram](image)

Fig. 7.2 Defining the direction of beams starting from a fruit.
A disadvantage of the polar coordinate system is the existence of singularities at the poles. Nevertheless, this effect is not significant if the angle resolution is small enough. The resolutions for the azimuth and the polar angles were therefore taken as $d\theta = \frac{2\pi}{1000} = 0.0063\text{rad} = 0.36^\circ$ and $d\varphi = \frac{\pi}{1000} = 0.0032\text{rad} = 0.18^\circ$. This resolution allowed resolving a branch with diameter 2 cm at distance 3 m, which was sufficient for this research.

To illustrate the cone search by the star map method, star maps were created for the following fruit: fruit number 16 and 7 of the TS apple tree model, and fruit number 40 of the CL apple tree model. The fruit are marked by circles in Fig. 7.3.

The corresponding star maps of the fruit are shown in Fig. 7.4. These are images of 1000 x 1000 pixels in size, where each pixel represents a direction of the rays starting from the fruit. The black pixels represent rays which do not reach obstacles. The gray-scale pixels represent rays reaching obstacles, where the intensity of the gray level represents the distance. The azimuth angle $\theta$ varies from 0 to 360° relative to the $X$ axis. The polar angle $\varphi$ varies from -90° to 90° relative to the $XY$ plane.

Fig. 7.3 Tall Spindle apple tree (left) and Central Leader apple tree (right) models with the fruit taken as examples for the reaching cone calculations. The considered fruit are marked by circles.
Fig. 7.4 Star maps for the considered fruit. Black pixels represent rays which do not reach obstacles. The gray-scale pixels represent rays reaching obstacles with the intensity of the color representing the distance according to the shown scale.

A binary segmentation was performed for the star maps to make the reaching cone calculation more convenient. A pixel has a value of 0 if it represents a ray not reaching obstacles or if the distance to a branch is more than some threshold, and has a value of 1 otherwise. The threshold is denoted as $d_{\text{max,inf}}$. An example of a binary map is given in Fig. 7.5.

The calculation of the star map is described in Algorithm 5.

**Algorithm 5.** Star map calculation.

**Input:** tree model, fruit number $k$.

**Output:** star map of fruit $k$.

1: Prepare array StarMap $1000 \times 1000$ $(\theta, \varphi)$, where $\theta \in [0..2\pi]$, $\varphi \in [-\pi..\pi]$

2: For each $i=1..1000$ and $j=1..1000$ and combination $(\theta_i, \varphi_j)$

3: For each branch on the tree

4: If the ray in the direction $(\theta_i, \varphi_j)$ collides with the branch

5: StarMap$_{i,j}$=distance between the fruit $k$ and the collision

6: Else StarMap$_{i,j}$=0

108
7.2. Reaching cone calculation

The calculation of a reaching cone for a fruit can be described by the following illustration using an umbrella. The tip of a straight closed umbrella is located at the fruit. Then the umbrella is gradually opened. The umbrella represents a cone with no obstacle inside it. If, during the opening, the umbrella collides with one or two obstacles at one or two points, it can continue to be opened, but now the umbrella shaft (the cone axis) tilts away from the collision points. The opening is ceased when the umbrella collides with a third obstacle limiting the shape of the umbrella, thus defining the opening angle and the axis of the reaching cone.

The described process was modeled with the help of the star map as follows: the direction of the shaft of the closed umbrella is depicted by a point located in a collision-free direction (black pixel in the star map). Since the initial directions of the umbrella or the initial points in the star map are unknown, a number of initial guesses are used to ensure that all possible reaching cones are derived. It was found experimentally, that $6 \times 6 = 36$ uniformly distributed initial guesses, presented in Fig. 7.5a, were sufficient to find all of the reaching cones.

![Fig. 7.5 Binary star maps. The grid of 36 uniformly distributed initial guesses, represented by red asterisks (a). The projection of the umbrella base circle, in blue (b), at the beginning of the umbrella opening at one of the initial guesses.](image)
The circular base of the umbrella is projected on the star map to find the relative position of the umbrella and the obstacles. One of the initial directions in Fig. 7.5a was taken as the shaft of the umbrella, while the circular base of the umbrella was projected to the curve around the initial point in Fig. 7.5b. During the umbrella opening, the circle of the umbrella’s base increases and its projection in the star map increases (Fig. 7.5a). Collision of the umbrella with one or two obstacles corresponds to the collision of the projected circle with the points corresponding to the directions with obstacles (white pixels in the star map). After the first collision (Fig. 7.6a), the projection of the circle continues to increase, but the circle center (the initial point) moves in the map away from the collision point in the direction shown by the arrow in Fig. 7.6b. Similarly, after the second collision, the circle centers moves in a direction opposite to the average of both collision points (Fig. 7.6b, c). The enlargement of the circle projection is stopped when the circle projection is limited by three collision points (Fig. 7.6d). The size and center of the resultant circle projection define the opening angle and the direction of the reaching cone. The height of the cone is defined by the farthest collision point.
Fig. 7.6 The process of umbrella opening modeled by the increase and movement of the projection of the umbrella base circle. (a) Increasing the base projection until collision with an obstacle. (b) Increasing the base projection touching the obstacle until collision with a second obstacle. (c) Increasing the base projection touching two obstacles until collision with a third obstacle. (d) Collision of the base projection with three obstacles defines the base projection size and location in the map and the umbrella opening angle and direction. The arrow shows the direction of the circle projection movement while it increases.
Fig. 7.7 Views of a reaching cone for fruit 16. The cone vertex is located in the center of the fruit. The cone surface is defined by three points of collision with obstacles: tree stem, trellis wire and one of the tree branches.

A number of views of the resultant reaching cone are presented in Fig. 7.7. The cone vertex is located in the center of the fruit. The cone axis is represented by a blue line. The cone surface is defined by the collisions with the tree stem, trellis wire (almost included in the cone surface) and one of the branches. The cone has a wide opening angle, making it almost planar and the cone axis is short, since there are no obstacles in this direction.

The described process of cone finding was performed for all initial guesses. A large number of initial guesses can yield reaching cones. As a result, redundant overlapping cones could be found. They were sifted according to the following rules.

1. Cones with an opening angle less than 30° are discarded.
2. The difference between two cones is defined in Fig. 7.8 as the angle covered by only one cone. If the smaller difference is less than 30% of the opening angle of the wider cone, the narrower cone is included into the wider cone, and is discarded.

Fig. 7.8 Definition of the difference between two cones.
These rules leave about 10 cones per fruit, maintaining, in most cases, a sufficient level of detailing of the obstacle-free areas by the reaching cones. However, there are obstacle-free areas that are not characterized by the reaching cones. Hence, the characterization is conservative, which sometimes leads to calculating the robot motion trajectories based on reaching cones that are longer than the actual robot trajectories.

All of the non-overlapping cones of fruit 16 are presented in Fig. 7.9.

The resultant non-overlapping cones of fruit 7 of the TS apple model, and fruit 40 in the CL apple model are presented in Fig. 7.10 and Fig. 7.11, respectively.

Fig. 7.9 Seven reaching cones represented as the projections of the cones’ base circles in the star map of fruit 16 and as cones in the Cartesian coordinate system.
Fig. 7.10  Map of the reaching cone projections for fruit 7 on the Tall Spindle apple model and isometric views of the cones.
Fig. 7.11  Map of the reaching cone projections of fruit 40 in the Central Leader apple model and isometric views of the cones.
The process of the reaching cones calculation is described in Algorithm 6.

**Algorithm 6.** Reaching cones calculation.

**Input:** tree model.

**Output:** reaching cones for all the fruit on the tree.

1: For each fruit on the tree
2: Create the star map of the fruit by the Algorithm 6
3: **For** each initial direction of reaching cone
4: **While** the number of collisions with the branches < 1
5: Increase the cone opening angle
6: **While** the number of collisions with the branches < 2
7: Increase the cone opening angle and move the cone axis from the collision point
8: **While** the number of collisions with the branches < 3
9: Increase the cone opening angle and move the cone axis from the average of the collision points
10: //sifting
11: **If** the opening angle of the last calculated cone < 30°
12: Discard the last calculated cone
13: **For** all calculated cones
14: **If** the cone overlaps with the last calculated cone
15: Add the cone with the biggest opening angle
16: **Else** add the last calculated cone
7.3. Reachability map

The reaching cones represent a geometrical feature of the agricultural environment characterizing the obstacles. They can be used to make decisions about the robot location before the robot optimization.

A number of reachable fruit can be found for each point in the workspace close to the tree. We divided the workspace around the tree using a 3D grid defined by the following parameters: height $z$, distance $r$ to the $Z$ axis (approximately corresponding to the tree trunk), and direction $\theta$ around the $Z$ axis relative to the $X$ axis as shown in Fig. 7.12.

The percentage of reachable fruit is found for each vertex of the grid around the tree. The 3D table of the reachability percentage from each grid vertex is defined as the reachability map. To show an example of a reachability map in 2D, we take a slice of the reachability map with a fixed distance to the tree $r = 1$ m. The surface of the reachability map for the CL apple tree model is shown in Fig. 7.13 in up, front and side views. The percentage of reachable fruit has a large variance between different locations around the tree.

![Fig. 7.12 Coordinate systems defining the location of a point around the tree model.](image)
Fig. 7.13  Reachability map for the Central Leader apple tree model with fixed distance $r = 1$ m. The values of the reachable fruit percentage strongly depend on the location around the tree.

The highest values are achieved at the locations with $\theta = 180^\circ$ (Fig. 7.13b), corresponding to the location in front of the tree trunk. The lowest values are achieved at $\theta \sim 90^\circ$ and $\theta \sim 310^\circ$ (Fig. 7.13b), corresponding to the location inside the tree row. The height with the highest values of reachable fruit percentage is $z \sim 0.75$ m (Fig. 7.13c). According to this map, there is a location of a point from which 76% of the fruit can be reached.

By choosing a number of points in the workspace, an even higher percentage of reachable fruit can be achieved. Finding the optimal combination of points is a complex optimization problem, including $3 N_{loc}$ optimization variables, where $N_{loc}$ is the number of point locations. It can hardly be solved exactly, and thus in these examples, near-optimal solutions were found with the help of a GA. The dependence of the total percentage of reachable fruit on the number of point locations $N_{loc}$ is illustrated in Fig. 7.14b. According to the graph, four locations are sufficient to have a direct line of site to all of the fruit on the tree, and two locations are sufficient for 95% of the fruit. This can be used to approximate the number of locations for robot with specific kinematics sufficient for picking the required amount of fruit and the robot kinematics.

The near-optimal locations for $N_{loc} = 1, 2, 3, 4$ are shown in Fig. 7.14a. For a robot with structure PRRP, the reachability analysis gives an exact number of robot locations. Assuming that the robot base is located at one of the points, the RR joints enable the robot to point along the ray connecting the robot base and the fruit. The last P joint enables motion along the ray to the fruit. The first P joint adjusts the height of the robot base location point.
Fig. 7.14  Dependence of the total percentage of reachable fruit on the number of point locations $N_{loc}$. The near-optimal locations are presented by the asterisks in (a). The graph of the dependence is shown in (b).
8. Environment Characterization by Reaching Cones

Task-based robot optimization for environments with a large number of objects takes a long time as discussed in Section 4.7. The most complicated environment for which the optimal robot was found in Section 4.8 had 144 fruit and 171 branches. In that case, the average cost-function evaluation time was about 10 minutes on a computer Intel® Core™ i5-2310 CPU 2.9GHz, 4GB RAM. For a more complicated environment, for example, with 297 fruit and 379 branches, the average cost-function evaluation time is about 20 hours. A solution to that problem cannot be achieved in an acceptable time on a regular computer.

The reaching cones are used to speed up the calculations in the following stages:

1. The solution of the robot navigation problem found during the evaluation of the robot performance cost function is replaced by navigation inside the reaching cones.

2. In the optimal robot location problem, the number of location combinations depends strongly on the number of possible robot locations, resulting in combinatorial explosion. The possible robot locations are replaced by a lower number of suitable robot locations which allow the robot to reach a number of fruit with relative ease. The suitable robot locations are found with the help of the reachability map (Section 0) built using the reaching cones, as described in detail in Section 8.2.

3. Optimization based on the CE achieved by the fruit-clustering method, described in Section 6.7 can be performed only assuming the absence of branches, which is not practical in the general case. To include the influence of the robot obstacles (i.e. branches) during the robot optimization, the CE is built by combining the fruit-clustering method and the reaching cones.

8.1. Navigation inside the reaching cones

Robot motion planning is the problem of finding the robot trajectory in obstacle-free areas. To find the robot trajectory, we do not need to consider the obstacles, but the free areas, which are not occupied by the obstacles. These free areas are characterized by the reaching cones of the
fruit. To use the reaching cones for simplification of the robot navigation, we assumed that if a robot of any structure can move inside a reaching cone without collisions, its trajectory inside the cone can be a straight line in the robot configuration space (representing the fastest trajectory).

Based on this assumption, we formulated Algorithm 7 for finding the robot trajectory in a reaching cone, replacing the solution of the robot navigation problem from the home configuration of the robot to a specific fruit.

**Algorithm 7.** Navigation inside the reaching cones for a given fruit, robot kinematics, location and home configuration.

**Input:** fruit with all its reaching cones, robot kinematics and location

**Output:** robot trajectory in the robot joint space

1: Solve the inverse kinematics for reaching a given fruit
2: For all reaching cones of the fruit
3: If there exists a robot configuration allowing it to reach the fruit AND is included in the reaching cone
4: If the robot home configuration is included in the reaching cone
5: The robot trajectory is the straight line in the robot joint space connecting the home configuration with the fruit-reaching configuration
6: Else
7: Find the point on the reaching cone closest to the robot's end effector in the home configuration and solve the inverse kinematics for reaching this point
8: The robot trajectory consists of two straight lines in the robot joint space connecting the home configuration with the point on the cone base, and the point on the cone base with the fruit-reaching configuration
9: Else
10: No trajectory exists
11: Among all found trajectories choose the shortest

The reaching cones do not provide an exact description of the obstacle-free areas, but provides a conservative characterization of those areas. Furthermore, since every fruit has a number of reaching cones, the overall number of cones can exceed the number of branches. Thus, a
characterization of the environment using the reaching cones to solve the robot navigation problem may actually require consideration of a number of objects greater than the number of objects in the AE. However, Algorithm 7 provides a method for handling the reaching cones while solving the navigation problem which is more efficient than considering the actual branches, eventually leading to a reduced runtime.

To check the effectiveness of the characterization in terms of cost function and runtime, we compared the cost-function values of a robot navigating in the AE with the robot navigating in the environment with obstacles characterized by the reaching cones (Appendix Table B.3). Three robot structures (PPP, RRP and RRR) were checked for all three apple tree models (TY, YT and CL).

According to the results of the comparison, the difference in the cost function in most cases was below 10% and the percentage of unpicked fruit was below the allowed limit, while the advantage in computation runtime varied from 2 to 10, which is a significant decrease.

8.2. Robot location optimization using the reaching cones

The analysis of fruit reachability was used to improve the effectiveness of the solution to the robot location optimization problem. The method for robot location optimization developed in this research was described in Chapter 5. The method was based on creating the possible locations of the robot around the tree and finding their optimal combination providing the most effective fruit reaching. The area of distribution of the possible robot locations was defined according to the tree dimensions: maximal and minimal fruit height and maximal and minimal canopy radius. The possible locations were distributed uniformly in concentric circles with the tree trunk in the center (Fig. 8.1a) or parallel to the tree row (Fig. 8.2a). All combinations of locations for a given number of robot locations $N_{loc}$ were checked, and the optimal combination was found. The solution using this method can take a long time (discussed in Section 4.7) for a complicated tree or a large number of robot locations.

To decrease the time to the optimal solution for the robot location, the number of possible robot locations can be decreased by the analysis of fruit reachability developed in Section 0. A point on the ground plane is suitable for the robot location if, from the points above the robot location, a large number of fruit are reachable, and if the points are located close to the fruit. These two
criteria are combined into a measure for robot location suitability, depicted as $\varphi_i$ for the point $(x_i, y_i, 0)$. The robot location suitability must be higher for larger numbers of reachable fruit and for shorter distance to the fruit. Hence, robot location suitability is calculated by the formula

$$\varphi_i = \frac{N^{2}_{\text{reachable fruit}}}{d}$$

Eq. 8.1

where $N_{\text{reachable fruit}}$ is the number of the fruit reachable from the point $(x_i, y_i, 0)$, and $d$ is the distance to the closest reachable fruit. The influence of either criterion on robot location suitability can be adjusted by modifying the exponents of $N_{\text{reachable fruit}}$ and $d$. Eq. 8.1 was found experimentally to provide the best characterization for location suitability.

The robot location suitability $\varphi_i$ was calculated by the following algorithm.

**Algorithm 8.** Computation of the robot location suitability $\varphi_i$.

**Input:** tree model, possible robot location point $i$ $(x_i, y_i, 0)$ near the tree

**Output:** suitability $\varphi_i$ of point $i$

1. For some $n$, take $n$ points above the point $(x_i, y_i, 0)$ with heights $z_1, \ldots, z_n$ distributed uniformly across the height of the tree

2. For each point $(x_i, y_i, z_j)$ in the column, find all reachable fruit and the Euclidian distances to them

3. Find the total number of reachable fruit from the column

4. Calculate the value of the location suitability $\varphi_i$ for the point $(x_i, y_i, 0)$ according to Eq. 8.1.

According to Eq. 8.1, $\varphi_i$ does not have any physical meaning; hence, to be interpretable, all $\varphi_i$ are normalized by their maximal value. The values of $\varphi_i$ for the possible robot location points around the CL apple tree model are represented by dots in Fig. 8.1b. A surface approximating the $\varphi_i$ was built by the Radial Basis Function method and is shown in Fig. 8.1b. The closer the points are to the tree, the higher the $\varphi_i$. There are a number of local maxima close to the tree.
Fig. 8.1 Choosing the points suitable for the robot location among all possible robot locations for the Central Leader apple tree (a,b) and Y-trellis apple tree (c,d). The possible points are represented by dots, and the suitable points are surrounded by circles in (a) and (c). A surface approximating the location suitability is passed through the calculated $\varphi_i$ values in (b) and (d). The triangles in (a) and (c) represent the robot locations found while considering all possible robot locations, the squares represent the robot locations found based on the suitable robot locations.
To define the suitable robot locations, a threshold for the location suitability $\phi_i$ was used. The exact value of the threshold cannot be defined analytically. Using trial and error, it was found that a threshold of 0.1 gives acceptable results for the robot location optimization for different tree models. The points with $\phi_i > 0.1$ are marked by circles in Fig. 8.1b and were taken as the suitable robot location points for the robot location optimization. The values of $\phi_i$ around the YT apple tree model are presented in Fig. 8.1d, and for the YT apple tree row, in Fig. 8.2b.

![Diagram](image)

**Fig. 8.2** Choosing the points suitable for the robot location among all possible robot locations for a row of Y-trellis apple trees. The possible points are represented by dots, and the suitable points are surrounded by circles (a). A surface approximating the location suitability is passed through the calculated $\phi_i$ values (b). The triangles in (a) represent the robot locations found while considering all possible robot locations, the squares represent the robot locations found based on the suitable robot locations.
A robot with RRP structure and the CL apple tree model were used as an example for the robot location optimization. The robot location optimization was performed by the algorithm developed in Chapter 5, but now the number of possible robot locations was significantly decreased: the number of all possible locations was 114, while the number of suitable locations was 63. The optimal robot locations found from the suitable robot location points (depicted by squares in Fig. 8.1a) were compared with the optimal locations found within all possible points (depicted by triangles in Fig. 8.1a). The found optimal locations were close: two of them were coincident, and two were neighboring. The performance cost functions for the two robot locating methods were also close: 0.5522 for the suitable locations, and 0.5507 for all possible locations.

The runtime needed to solve the robot location optimization consists of the time for the inverse kinematics solution for each possible robot location for each fruit, and the search of the optimal location combination. The total time for the robot location optimization based on all possible points was 2723 seconds, where 227 seconds were needed for the inverse kinematics solution, and 2496 seconds for the optimal location combination search. The total time for the robot location optimization based on the suitable points was 160 seconds, where 132 seconds were needed for the inverse kinematics solution, and 28 seconds for the optimal location combination search. This time does not include the calculation of the reaching cones and the reachability map. The time required for this calculation is negligible and does not alter the overall result. The time ratio in the inverse kinematics solution depends on the ratio between the number of all possible locations and the number of suitable locations. The optimal location combination search is a combinatorial problem with a solution runtime that depends strongly on the number of points. Therefore, the decrease in the number of points led to a dramatic decrease in the runtime.

The robot location optimization based on all possible points and on suitable points is compared for a number of tree models and robot structures in Appendix A. According to the results of the comparison, the difference in the cost function in most cases was below 10%. The calculation time ratio increased with increasing environment complexity: from 2 for a simple tree, up to more than 100 for a row of five trees. Thus, the reachability analysis helped find the suitable robot locations, leading to a near-optimal solution with significantly shorter computation time.
8.3. Environment characterization by fruit clustering and reaching cones

The reaching cones, along with the fruit clustering developed in Chapter 6, are now used to construct a CE of the robot obstacles and robot targets. The advantages of both characterization methods were combined to achieve maximal simplification of the environment, thereby speeding up the solution of the robot optimization problem.

In the fruit clustering, the number of fruit was decreased by replacing a number of actual fruit by a single characteristic fruit. To decrease the number of reaching cones, a principle similar to fruit clustering was applied: replacing a number of reaching cones (included in a cluster of fruit) by a single characteristic reaching cone.

To achieve this replacement, the clustering must be performed not only in the space of the fruit location (which is a Euclidian space with X, Y and Z coordinates), but in the extended space that also includes the parameters of the reaching cones (the cone opening angle, two parameters defining the direction, and cone height). Clustering in this space with seven dimensions is computationally impractical (particularly because of the number of initial guesses). Hence, clustering of the fruit location and the reaching cones was separated and performed in two steps, as described in Algorithm 9.

First, the fruit in the environment were divided into clusters with a given maximal size $D_{clust}$ similar to the clustering without obstacles (Section 6.7). The extreme characteristic fruit were found for each cluster. Second, the reaching cones were characterized in each cluster. The number of fruit in each cluster was decreased according to the similarity of their reaching cones.
Algorithm 9. Environment characterization by clustering and extreme target.

**Input:** Given AE, $a_{allow}$, $D_{clust}$, and reaching cones for all fruit

**Output:** CE including characteristic targets with their reaching cones

1: Calculate the $N_{clust}$ according to $D_{clust}$

2: Divide the targets of the AE into $N_{clust}$ clusters

3: For each cluster

4: Define the characteristic extreme targets

5: For each fruit $i$ in the cluster

6: For each fruit $j \neq i$ in the cluster

7: If the reaching cones of fruit $i$ and fruit $j$ overlap

8: Delete fruit $i$ and fruit $j$

9: Create a fruit with location between fruit $i$ and fruit $j$ and with minimal number of reaching cones of fruit $i$ and fruit $j$ covering all reaching directions for fruit $i$ and fruit $j$

The overlap was checked as in Section 7.2. All fruit remaining in the environment represent the characteristic fruit, characterizing both the fruit location and the fruit reaching cones.

An example of the environment characterization is presented for the CL apple tree model. The maximal cluster size $D_{max,clust} = 0.5$ m is taken from the conclusions drawn in Section 6.7 dividing the environment into five clusters with 10 extreme fruit. The clustering of the reaching cones results in 61 characteristic fruit, replacing 144 actual fruit; 171 branches are characterized by 394 reaching cones, with 6.5 cones for a characteristic fruit on average. The characteristic fruit are presented in Fig. 8.3b with their biggest characteristic reaching cones.
Fig. 8.3 Central Leader tree model (a) and its characteristic environment constructed by clustering and reaching cones for the Central Leader model.

8.4. Optimal robots based on the characteristic environment

The comparison of the RAE and RCE was similar to that performed in Section 6.7. The difference between the RAE and RCE is estimated in Table 8.1 by the proximity indices defined in Section 6.2.2: relative cost function difference $dF$, relative kinematics difference $dL$, and the percentage of unreachable fruit $\alpha_{uf}$. The runtime (in hours) is denoted by $T_A$ and $T_C$ for the optimization based on the AE and CE, respectively. The ratio between them is $TR = T_A/T_C$.

Similar to the previous examples in this chapter, high values of $dF$, exceeding 10% (marked by gray background), are related to the cases in which conservative characterization of the reaching cones leads to calculation of the longer robot motion trajectories. In most cases, $dF$ was below 10%. The calculation time advantage increases with increasing environment complexity: from 2.9 for a tree model including tens of objects, up to 15 for a tree model including hundreds of objects. Thus, using the CE allows finding the near-optimal robots with significantly shorter computation time.
Table 8.1 *Comparison of optimal robots based on actual and characteristic environments (RAE and RCE, respectively) by proximity indices for the three tree models.*

<table>
<thead>
<tr>
<th></th>
<th>PPP</th>
<th></th>
<th>RRP</th>
<th></th>
<th>RRR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$N_{loc}$</td>
<td></td>
<td>$N_{loc}$</td>
<td></td>
<td>$N_{loc}$</td>
</tr>
<tr>
<td>TS</td>
<td></td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>dF (%)</td>
<td>5.4</td>
<td>8.9</td>
<td>12</td>
<td>4.9</td>
</tr>
<tr>
<td></td>
<td>dL (%)</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>$\alpha_{uf}$ (%)</td>
<td>3.3</td>
<td>3.3</td>
<td>6.6</td>
<td>6.6</td>
</tr>
<tr>
<td></td>
<td>$T_A,T_C,T_R$</td>
<td>0.66, 0.2, 3.3</td>
<td>1.1, 0.26, 4.2</td>
<td>8.4, 2.9, 2.9</td>
<td></td>
</tr>
<tr>
<td>YT</td>
<td></td>
<td>3</td>
<td>3.7</td>
<td>4.6</td>
<td>8.7</td>
</tr>
<tr>
<td></td>
<td>dF (%)</td>
<td>44</td>
<td>7.7</td>
<td>37</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>dL (%)</td>
<td>5.9</td>
<td>7.4</td>
<td>4.4</td>
<td>4.4</td>
</tr>
<tr>
<td></td>
<td>$T_A,T_C,T_R$</td>
<td>12.3, 2.3, 5.3</td>
<td>16, 3.4, 4.7</td>
<td>15.4, 3.2, 4.8</td>
<td></td>
</tr>
<tr>
<td>CL</td>
<td></td>
<td>9.8</td>
<td>10</td>
<td>6</td>
<td>3.2</td>
</tr>
<tr>
<td></td>
<td>dF (%)</td>
<td>17</td>
<td>10</td>
<td>12</td>
<td>7.2</td>
</tr>
<tr>
<td></td>
<td>dL (%)</td>
<td>4.8</td>
<td>4.8</td>
<td>7.6</td>
<td>4.8</td>
</tr>
<tr>
<td></td>
<td>$T_A,T_C,T_R$</td>
<td>79, 5.2, 15</td>
<td>10, 2.3, 4.3</td>
<td>103, 15.3, 6.7</td>
<td></td>
</tr>
</tbody>
</table>

### 8.5. Optimal robot for complex environments

The comparison between the RAE and RCE is possible only for environments for which the RAE has been achieved. As was shown in Section 4.8, the search for the RAE for rows consisting of five trees takes an unacceptably long time. The approximated solutions can be found only using the CE. Assuming, that the characterization is successful, similar to the characterization in the previous examples, the difference in the cost-function value $dF$ is expected to be less than 10%.

The following tree row models are used as examples for complicated environments: YT apple row (YTR), CL apple row (CLR) and Nectarine row (NR). Each row includes five trees. The optimization runtime $T$ (in hours) and the performance cost function $F$ of the RCE in the AE are
presented in Table 8.2. If the RCE does not fulfill the optimization constraint $\alpha_{uf} < 5\%$, its $\alpha_{uf}$ is given. The empty cells represent cases in which $\alpha_{uf} > 15\%$. The total number of actual fruit in the row model, $N_{fruit}$, is given with the name of the row.

A number of RCE for the three considered environments are shown in the following figures: RCE with RRR structure located in 12 locations for the YTR (Fig. 8.6), RCE with RRP structure located in 12 locations for the CLR (Fig. 8.5), and RCE with RRP structure located in 12 locations for the NR (Fig. 8.6).

Table 8.2 Performance and optimization time for optimal robots based on a characteristic environment for complex environments.

<table>
<thead>
<tr>
<th></th>
<th>$N_{loc}$</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
<th>10</th>
<th>12</th>
<th>T, h</th>
</tr>
</thead>
<tbody>
<tr>
<td>YTR</td>
<td>PPP</td>
<td>1.8,</td>
<td>1.75,</td>
<td>1.63,</td>
<td>1.75,</td>
<td>2.1</td>
<td>1.93</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\alpha_{uf}=8%$</td>
<td>$\alpha_{uf}=9%$</td>
<td>$\alpha_{uf}=11%$</td>
<td>$\alpha_{uf}=9%$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RRP</td>
<td>1.3,</td>
<td>1.95,</td>
<td>2.24,</td>
<td>2.2,</td>
<td>2.00</td>
<td>2.07</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\alpha_{uf}=13%$</td>
<td>$\alpha_{uf}=20%$</td>
<td>$\alpha_{uf}=6.1%$</td>
<td>$\alpha_{uf}=6.7%$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RRR</td>
<td>3.08,</td>
<td>1.84,</td>
<td>1.12,</td>
<td>0.99,</td>
<td>0.69,</td>
<td>0.85</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\alpha_{uf}=15%$</td>
<td></td>
<td>$\alpha_{uf}=6.4%$</td>
<td>$\alpha_{uf}=6.4%$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLR</td>
<td>PPP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.31,</td>
<td>2.4</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$\alpha_{uf}=10%$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RRP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.13,</td>
<td>1.8,</td>
<td>1.89,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$\alpha_{uf}=9.8%$</td>
<td>$\alpha_{uf}=10%$</td>
<td>$\alpha_{uf}=6.2%$</td>
</tr>
<tr>
<td></td>
<td>RRR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.71,</td>
<td>1.8,</td>
<td>1.76</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$\alpha_{uf}=10.8%$</td>
<td>$\alpha_{uf}=6.8%$</td>
<td></td>
</tr>
<tr>
<td>NR</td>
<td>PPP</td>
<td></td>
<td></td>
<td></td>
<td>1.96,</td>
<td>2.08,</td>
<td>1.88</td>
<td>75</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$\alpha_{uf}=15%$</td>
<td>$\alpha_{uf}=9.5%$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RRP</td>
<td></td>
<td></td>
<td>1.89,</td>
<td>1.71,</td>
<td>1.72</td>
<td>1.57</td>
<td>82</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$\alpha_{uf}=12.7%$</td>
<td>$\alpha_{uf}=6.3%$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RRR</td>
<td></td>
<td></td>
<td></td>
<td>2.11,</td>
<td>2.23</td>
<td>2.14</td>
<td>2.05</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$\alpha_{uf}=9.4%$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Fig. 8.4 *Optimal robot based on a characteristic environment for the Y-trellis row.*

Fig. 8.5 *Optimal robot based on a characteristic environment for the Central Leader row.*
Fig. 8.6 *Optimal robot based on a characteristic environment for the Nectarine row.*

The results of this section represent the main achievements of this research corresponding to the third research objective (Section 1.2): the possibility of solving the robot optimization problem for complex environments in an acceptable time.
9. Orchard Architecture Design

Both the robot kinematics and the environment structure influence robot performance. To achieve optimal robot performance, both the robot and the environment must be optimally designed. In this chapter, we investigate the influence of the environment's geometrical features on the effectiveness of the robot designed to operate in this environment, find the existing tree architecture that is the best fitted to the robotic harvesting, and conduct a simulated experiment to design possible tree architecture optimally fitted to the robotic harvesting.

Modern high-plant-density training systems, such as the TS (Fig. 9.1b) and YT (Fig. 9.1c) were developed mainly to increase fruit yield and quality. In addition, they save on labor time during harvesting, providing a convenient environment for human harvesters. This advantage can also be used to provide an environment suitable to robotic harvesters, turning them into a profitable harvesting solution. In this chapter, we evaluate the fit of these training systems to robotic harvesting.

The agronomic and economic advantages of these tree-training systems, i.e., fruit quality, fruit yield and suitability for human harvesters, have been proven in agronomic research (Robinson et al., 1991). Hence, these aspects are not considered in the evaluation of the environment's fitness for robot performance.

Fig. 9.1 Three actual task environments and their models: apple trees trained by Central Leader (a), Tall Spindle (b) and Y-trellis (c) training systems.
To evaluate the environment's fit, we calculated the cost-function value $F$. The lower $F$ is, the more the environment is fitted to the robotic harvester. Throughout this chapter, fruit-picking time is evaluated using the AE without any type of environment characterization.

**9.1. Optimal robots for different environments**

A comparison of the different robot structures performing tasks in different environments, presented in Table 9.1, showed that each environment has its most effective robot structure, where the robot performance cost function (average picking time per fruit) $F$ has minimal values (shaded in gray): CL and YT environments have RRR, and TS environment has RRP. This can be explained by the differences in the geometrical features of the environments. Trees trained by the TS and YT methods constitute relatively structured environments. As a result, most of the fruit are surrounded by open space without obstacles, which enables the robot to approach the fruit by a straight line in the workspace from any robot base location. This type of motion is typical for the RRP robot structures. Trees shaped by the CL method have more fruit hidden behind branches which constitute obstacles to the robot's straight line motion. To approach them, additional revolute joint must be involved. The RRR robot is suitable for this type of motion.

**Table 9.1 Robot performance cost function $F$ for optimal robots for different environments.**

<table>
<thead>
<tr>
<th>$N_{loc}$</th>
<th>CL</th>
<th>TS</th>
<th>YT</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.48</td>
<td>1.9</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.43</td>
<td>1.1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1.97</td>
<td>0.40</td>
<td>0.95</td>
</tr>
<tr>
<td>8</td>
<td>1.87</td>
<td>0.40</td>
<td></td>
</tr>
<tr>
<td>RRP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.32</td>
<td>2.4</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.26</td>
<td>1.6</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1.13</td>
<td>0.25</td>
<td>1.45</td>
</tr>
<tr>
<td>8</td>
<td>1.12</td>
<td>0.25</td>
<td>1.45</td>
</tr>
<tr>
<td>RRR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1.35</td>
<td>0.41</td>
<td>1.24</td>
</tr>
<tr>
<td>4</td>
<td>0.87</td>
<td>0.28</td>
<td>0.84</td>
</tr>
<tr>
<td>6</td>
<td>0.71</td>
<td>0.28</td>
<td>0.81</td>
</tr>
<tr>
<td>8</td>
<td>0.61</td>
<td>0.28</td>
<td>0.81</td>
</tr>
</tbody>
</table>
9.2. Evaluation of total robot performance time

Till now we considered harvesting of a single tree without movements between the robot base locations. To understand the influence of movement time, $T_{mov}$, we consider the $T_{mov}$ in the interval [0 40]sec, and evaluated the total time, $T_{tot}$, needed to pick 7920 fruit in orchards shaped by CL, TS and YT training systems.

We assume that the trees in the orchard are the modeled trees located along the rows. The number of fruit is 144, 30 and 66 on the CL, TS and YT trees respectively. Hence, to model 7920 fruit, the row must include the following number of trees $N_{tree} = \frac{7920}{N_{fruit}}$, which is 55, 264 and 120 for the CL, TS and YT orchards respectively. The total time for the fruit picking is defined as:

$$T_{tot} = 7920 \cdot T_{fruit} + T_{mov} \cdot N_{tree} \cdot N_{loc}$$  \hspace{1cm} \text{Eq. 9.1}

where $T_{fruit}$ is the average time for the fruit picking equal to the cost-function value $F$. The total time for different $N_{loc}$ and robot types is presented in Fig. 9.2.

The optimal training system is defined by the minimal total time $T_{tot}$, which is a tradeoff between the average fruit picking time and the number of movements. For $T_{mov}$ close to zero, the optimal training system is TS with RRP robot located in $N_{loc} = 6$ locations. $T_{mov}$ less than 14 seconds is small enough to make the TS training system with $N_{loc} = 2$ the optimal using the advantage of TS small size. For $T_{mov}$ more than 14 seconds the density of TS becomes a disadvantage making the number of trees and movements too high. Then CL with low tree density becomes optimal.
Fig. 9.2 Total fruit picking time for different orchard types, robot types and number of locations depending on the movement time $T_{\text{mov}}$.

9.3. Preliminary environment design

Despite the complexity of the tree geometry, there are parameters that can be controlled by growers and changed in the tree models. One of them is the angle of the trellis tilt depicted by $\gamma_{\text{trellis}}$. We use $\gamma_{\text{trellis}}$ to achieve a tree model shaped by the YT system optimally fitted to the robotic harvesting.

This tree optimization is performed for $\gamma_{\text{trellis}}$ values between 5° and 85°, at steps of 5°, assuming that the other tree geometry parameters are independent of $\gamma_{\text{trellis}}$.

Three tree models with trellis tilt angle, $\gamma_{\text{trellis}}$, of 5°, 30° and 85° are shown in Fig. 9.3. The tree model with $\gamma_{\text{trellis}} = 5^\circ$ resembles the TS training method. The model with $\gamma_{\text{trellis}} = 30^\circ$ is close to the common YT trees. The model with $\gamma_{\text{trellis}} = 85^\circ$ resembles the trellises of kiwi vines.
Fig. 9.3  Three models of trees shaped by the Y-trellis method with different trellis tilt angle: 5° (a), 30° (b) and 85° (c).

The cost function $F$, depending on the Y-trellis tilt angle $\gamma_{\text{trellis}}$, is presented in Fig. 9.4. RRP and RRR robot structures are considered. For each structure, the optimal robots were found for the number of locations around the tree, $N_{\text{loc}}$, equal to 2, 4, 6 and 8.

The graphs in Fig. 9.4 describe the relationship between the environment and its optimal robot. For the RRR robot structure (Fig. 9.4a), the larger the trellis tilt angle, the lower the robot performance cost function, meaning that the environment is better fitted for robotic harvesting. If agronomic and economic considerations do not permit high tilt-angle values, 15° is optimal for the low angle values, and 35° is optimal for the middle angle values. In addition, locating the robot in two locations around the tree ($N_{\text{loc}} = 2$, one on each side of the tree row) causes relatively high values of the cost function. The unordered behavior of the graph for $N_{\text{loc}} = 2$ is a result of an insufficient number of robot locations: when a robot has only one location on each side of the row to pick fruit from one of the branches of the YT, it can only effectively pick fruit on the nearest
side of the branch, whereas the fruit from the other side of the branch are picked ineffectively, in contrast to cases where $N_{loc}$ is greater than 2.

The RRP robot structure (Fig. 9.4b) with $N_{loc} = 2$ is efficient only with low values of $\gamma_{trellis}$, where the Y-trellis is similar to a fruit wall. For $\gamma_{trellis} > 40^\circ$, there are no solutions (none of the robots fulfill the constraint). The reason for this is the same as the reason for the unordered behavior of the graph for $N_{loc} = 2$ for RRR: inability of a robot in a single location on each side of a row to pick fruit from both sides of a branch. The fit of the environment decreases for $N_{loc} = 4$, and remains almost constant for $N_{loc} = 6$ and $N_{loc} = 8$ when the tilt angle grows. Thus, for the RRP robot structure with $N_{loc} = 2$ and $N_{loc} = 4$, the most effective tilt angle for all values of $N_{loc}$ is close to zero, producing a fruit wall. For a greater number of locations, the tilt angle has little effect on the cost function.

Assuming that any $\gamma_{trellis}$ angle can be achieved, the optimal combination is the Y-trellis with $\gamma_{trellis} = 85^\circ$ and a (RRR) robotic arm located in at least six locations around a tree.

Three approaches were shown in this section: checking the existing tree-training methods, considering the motion time of the robot platform, and simulation of trees. All of these approaches should be used during a simultaneous design of the optimal robot and its environment to achieve the minimal cost of fruit harvesting. Ideally, the full simultaneous design, including all important agronomic and economic factors, must be performed.
Fig. 9.4 Performance cost function of the optimal robots with RRR (a) and RRP (b) structures for Y-trellis tree models. The trellis tilt angles are in the range [5°..85°]. The number of robot locations around the tree, \( N_{\text{loc}} \), is 2, 4, 6 and 8. Starting from 45° for the RRP robot, \( N_{\text{loc}} = 2 \), there are no solutions for the optimal robots fulfilling the optimization constraint.
10. Discussion

In this research, we explored a number topics related to agricultural robotics. These topics dealt with the optimal design of the robot tailored to its environment and the design of the environment to improve the robot’s effectiveness.

11.1. Robot task-based optimization

Fitting the robot’s kinematics to its working environment is critical in the robot-design process. If a robot is designed for a specific task, such as an agricultural task, the design process comes down to task-based robot optimization, which takes into account all of the relevant details of the task and working environment. Therefore, these details of the actual agricultural environment must be thoroughly modeled. In this research, we used a simple and reliable, but time-consuming method for the plant modeling. When designing a robot for a specific farm, the method proposed in the Future Work section can be used.

Researchers and commercial developers of agricultural robotics usually fit the kinematics of the designed robots to their task. However, the optimal fitting of robots based on a detailed modeled environment can have a significant advantage in the performance cost function relative to other robots, such as non-optimally fitted robots or universal industrial robots.

11.2. Robot location optimization

The robot location optimization problem is an important sub-problem of robot optimization. Robot location optimization can be too complicated for the agricultural environment. In studies of agricultural robotics, the optimal robot location is usually assumed. To solve this problem with the help of common optimization methods, the problem must be formulated in a simplified way, taking into account the known features of the problem.

The developed location optimization method finds the global optimal solution with a given precision in an acceptable time. In this research, we applied a simple grid search algorithm for the optimization. This application can be generalized to more effective and accurate optimization algorithms, such as the simplex method.
11.3. Environment characterization

Environment characterization is a method developed to simplify the relatively complicated problem of robot optimization for agricultural tasks. Researchers performing task-based robot optimization usually simplify the task, losing important details in the process and influencing the result. We propose to characterize an environment by the characteristic environments (CE), which is constructed based on the actual environments (AE), and keeps all of the geometrical features needed for the robot optimization.

In the preliminary stage of development, only fruit features were used. The robot optimization based on characteristic environments (RCE) was compared with the robot optimization based on actual environments (RAE). In most examples, the RCE was close enough to the RAE (differing by less than 10%). The ratio between the runtimes for finding the RAE and RCE ranged from 3 to 100. The time advantage became significant for trees with a large number of fruit.

The idea of reaching cones was developed to characterize the plant branches and other robot obstacles. The reaching cones were used to analyze the ability of the designed robot to reach its target in obstacle-free areas. The main advantage of the reaching cones is the fact that they are objects in the workspace and do not depend on the robot. This is necessary for analysis during the robot optimization, since the robot itself is unknown and its configuration space cannot be created. Robot optimization based on the CE constructed by the reaching cones is advantageous in complex cases, when the optimization based on the AE cannot be accomplished in an acceptable time (in some cases, taking more than a week).

Analysis of fruit reachability with the help of the reaching cones gives information about the number of robots needed to reach the required number of targets, and the kinematic structure of the robot. This information helps make decisions about the kinematics of the robot intended to work in a given environment. Finding the possible robot locations by fruit reachability decreases the time taken to solve the robot optimal location problem.

11.4. Environment design

The design of the agricultural environment is a complicated problem that includes economic and agronomic aspects. To solve it, we evaluated the fit of the existing tree architectures produced by different tree-training systems for robotic harvesting. The comparison showed that the harvesting
time needed for high-density training systems, such as TS, is less than that needed for the conventional tree-training systems. Therefore, this training system is better suited to robotic harvesting. In addition to the agronomical advantages of high-density training systems, they provide automation advantages.

We made a preliminary design of the optimal environment by changing one of the agronomic parameters, i.e., the branching angle of the leading branches in the YT training method. The possibility and cost of growing a plant with a given branching angle is the agronomic aspect, which is outside the scope of this research. The robot optimization based on the simulated trees with different values of branching angle gave an index for effectiveness of different tree shapes.

High-density training systems are based on growing tall trees at high density and eliminating small branches. This provides two main advantages. Concentration of the fruit in a compact volume (near the row plane for TS and along the trellis for YT) allows shortening of the robot’s link lengths and enables the robot to shorten its time of movement between locations around the tree. In addition, the lower number of branches provides a working volume with minimum obstacles. An additional non-mechanical advantage of the high-density training systems is the decreased number of branches and leaves occluding the fruit, which can simplify fruit recognition and decrease the robot's trajectory planning time.

11.5. Optimization cost function

Optimization of the cost-effectiveness of an agricultural robot is a multidisciplinary problem involving economic considerations that depend on time and country, agronomic considerations that depend on plant behavior, and engineering design intended to fulfill all of the farm requirements. To solve the problem in the general case for the purpose of research, we had to make a number of assumptions and simplifications. In this research, we reduced all of the economic factors to the average fruit picking time, which is one of the most general and important factors in industrial and agricultural processes. The developed ideas and methods can now be used to solve the problem for a specific farm.
11. Conclusions and Future Work

The objectives of this research, defined in the Introduction, were achieved as follows.

The **first objective** of the research was to produce an exact and sufficient description of the actual agricultural tasks and their environments. A library of the plants representing the working environment for the robot was created. The library has 20 models of fruit trees, which were used to develop the methods for the robot and its environment optimization.

The **second objective** of the research was to show the importance of the methodological search for a robot that is optimally fitted to its task, and use of the exact task model. The optimal robots were found by task-based optimization for different environments and compared with other robots that were able to work in these environments.

The **third objective** of the research was to develop a methodology to help solve the robot optimization problem in an acceptable time with sufficient precision. The methodology for the environment characterization toward robot optimization was checked by comparison with optimization based on AE. In the complex cases, this methodology yielded near-optimal results in an acceptable time, whereas the optimization based on the AE did not succeed.

The **fourth objective** of the research was to show the significance of the influence of the robot environment on the robot cost, and create a preliminary design of the agricultural environment. A number of existing environments were tested for operation with robots, and the most fitted environment was found. A number of simulated trees with different branching angles were tested, and the tree most fitted to the robotic operation was found, indicating the ability to improve the effectiveness of the environment by growing and shaping the trees with optimal architecture.

Future research can be conducted in the following areas.

12.1. Number of trees

The goal of the robot optimization is to find the optimal robot for performing its task in a specific environment, for example, in an orchard with trees of the same type shaped by the same training method. We solved the robot optimization problem based on a group of trees representing the entire orchard. Analysis of the similarity of the environment's geometrical features showed that
the differences in tree structures from one orchard can be high. Thus it may be that a single tree or a small group of trees cannot represent the entire orchard. To achieve an optimal robot for an entire orchard, the orchard must be characterized by a number of trees, sufficient to represent all of the features of the orchard influencing the design of the robot.

The sufficient number of representative trees can be found in future work. To find this number for an orchard, a significant part of the orchard (more than five trees) must be modeled. Then the minimal group of trees yielding a robot similar to the optimal robot for all modeled trees must be found.

12.2. Reaching cones

The method of environment characterization by reaching cones can be further developed. The reaching cone is a preliminary method of describing obstacle-free areas for the robot's motion. The parameters of the reaching cones or the shape describing the obstacle-free area can be improved to increase the effectiveness of the characterization.

Characterization by the combination of reaching cones and fruit clustering can be improved and further developed. The effectiveness of the characterization can be increased by clustering, in an enlarged space, features of: fruit location, fruit orientation, reaching cone features, etc.

12.3. Self-adjusting robot

To achieve the optimal robot intended for work in some orchard, the orchard must be modeled (by statistical sampling) and the optimization process performed. One of the directions of future studies might be the replacement of these time-consuming actions by a specially designed robot. We define this robot as a self-adjusting modular robot. The joints of this robot have to be manually changeable by a robot operator: joint type, angle of the joint axis and lengths of the robot links. This robot can start to perform its task during modeling of its environment. After achieving a sufficient amount of information about the environment, the optimal robot kinematics is found, and the robot operator improves the robot, similar to a closed-loop control. Thus, the optimal kinematics of the robot is found during the work, and the robot adjusts itself to the environment, thereby maintaining maximal effectiveness.
12.4. Environment design

Part of the environment design consists of defining the tree parameters, such as branch length, branching angle, number and distribution of the branches and fruit. These parameters can be defined only by agronomical description of the plant. Future work on environment design must combine the description of robot kinematics with that of plant geometry and limits in plant training.

Trellis cables and supports also represent obstacles for the robot. As part of the environment design, some engineering recommendations defining the positioning of these supports can be made in a way that minimizes their interference with the operation of the robot.

Methods of describing the environment, such as those based on L-systems, can be used for an optimization of the environment best suited to robotic harvesting. An exact description of the tree geometry, topology and stochastic behavior is complex and demands finding the numerous parameters of L-systems from field experiments. Future studies might use the tree models to find these parameters for trees in a given orchard.
Appendix A. Environment Library

The library consists of a description of the original plant and its image, method of modeling, and front, side and isometric views of the plant model.

A.1 Tall Spindle apple tree model (Apple_1_TallSpindle, TS)

An apple tree of the type McIntosh was trained by the TS system in Massachusetts in 2010. The tree was reconstructed from the picture by measuring the object’s location on the picture plane and assuming the depth.

Fig. A.1 Actual plant.

Fig. A.2 Model of Apple_1_TallSpindle.
A.2 Central Leader apple tree models (Apple_1_7_14, CL1, CL2, CL3, CL4, CL5)

The following five trees were trained by the CL system in a commercial orchard belonging to the Fridman family in Nov, Golan Heights on 1 Jul 2014. The trees were modeled by the measuring device.

Fig. A.3 Actual plants CL1, CL2 and CL3.

Fig. A.4 Actual plants CL4 and CL5.
Fig. A.5 *Models of LC1, CL2 and CL3.*

Fig. A.6 *Models of LC4 and CL5.*

A row of five trees Apple_1_7_14Row was constructed with the tree models.

Fig. A.7 *Model of Apple_1_7_14Row.*
A.3 Central Leader apple tree models (Apple_11_10_13, Apple_30_10_13)

The following two trees were measured in an experimental orchard in Ramat Matityahu, Upper Galilee. The trees were modeled by the measuring device.

![Actual plants Apple_11_10_13 and Apple_30_10_13.](image1)

Fig. A.8 Actual plants Apple_11_10_13 and Apple_30_10_13.

![Models of Apple_11_10_13 and Apple_30_10_13.](image2)

Fig. A.9 Models of Apple_11_10_13 and Apple_30_10_13.
A.4 Y-trellis apple tree models (Apple_YTrellisL, YT1…5, YTA5…A85)

The tree was reconstructed from the picture. The relatively simple structure of the tree allows formulating the rules of the L-systems. All models were built according to this rule.

Fig. A.10  Actual plant Apple_YTrellisL.

L-systems description

The models of the trees trained by the YT method were built with the help of the L-systems method (Prusinkiewicz and Lindenmayer, 1990). The tree geometry was defined by the following L-systems rules (symbols defined in Prusinkiewicz and Lindenmayer, 1990). The parameters of these rules were achieved by analyzing the pictures of trees growing in the orchards.

\[ n = 3 \]

```plaintext
#define \( \alpha_1 \) 30° /* \( \gamma \)trellis */
#define \( \alpha_2 \) 45° /* branching of the generations 2 and 3 */

w: \( A[(\alpha_1)FFFFF][(180°)\&(\alpha_1)FFFFF] \)

p₁: \( F \rightarrow A[+(\alpha_2)F][-\(\alpha_2\)F] \)
```

In the designed L-systems, the number of branch generations is three \((n = 3)\). The first branching angle is defined as 30°, and the second as 45°. The axiom \(w\) builds the following structure: build vertical branch A (the tree trunk), turn around the X axis on 30° \(\&(\alpha_1)\), build five branch intervals \(FFFFF\), return to the previous position \(]]\), turn around the Z axis on 180° \(\&(180°)\), turn around the X axis on 30° \(\&(\alpha_1)\), build five branch intervals \(FFFFF\). The rule \(p₁\) replaces each branch interval \(F\) by the following structure: build branch interval A, turn around the up
direction on 45° and build one branch interval (+α₂F), return to the previous position (||), turn around the up direction on -45° and build one branch interval (-α₂F).

Fig. A.11  Models of Apple_YTrellisL and tree row consisting of five models created by the L-Systems rule.

Fig. A.12  Models of Apple_YTrellisL with trellis tilt angles 5°, 30° and 85° created by the L-Systems rule.
A.5 Nectarine tree models (Nectarine_30_6_14, N1, N2, N3, N4, N5)

The following five trees were measured in the Fridman family commercial orchard in Nov, Golan Heights, on 30 Jun 2014. The trees were modeled by the measuring device.

Fig. A.13 Actual plant Nectarine_30_6_14.
Fig. A.14 *Models of N1, N2 and N3.*

Fig. A.15 *Models of N4 and N5.*

A row of five trees Nectarine_30_6_14Row was constructed with the tree models.

Fig. A.16 *Model of Nectarine_30_6_14Row.*
A.6 Nectarine tree models (Nectarine_6_4_14, Nectarine_30_3_14)

The following four trees were measured in an experimental orchard at the Volcani Center in Bet Dagan. The trees were modeled by the measuring device.

Fig. A.17 Actual plant Nectarine_6_4_14 and Nectarine_30_3_14.

Fig. A.18 Models of Nectarine_6_4_14 and Nectarine_30_3_14.
A.7 Peach tree models (Peach_23_03_14, Peach_24_03_14)

The following trees were measured in an experimental orchard at the Volcani Center, Bet Dagan. The trees were modeled by the measuring device.

Fig. A.19  Actual plants Peach_23_03_14 and Peach_24_03_14.

Fig. A.20  Models of Peach_23_03_14 and Peach_24_03_14.
A.8 Tangerine tree models (Tangerine_20_3_14)

The tree was measured in a commercial orchard. The tree was modeled by the measuring device.

Fig. A.21 Actual plant Tangerine_20_3_14.

Fig. A.22 Actual plant Tangerine_20_3_14.
Appendix B. Analysis of the Characterization

B.1 Example in Section 6.3.1

To analyze the characterization by the mean characteristic targets, we use 100 random actual environments (AE) with 12 targets from the domain $1 \leq x \leq 2$, $1 \leq y \leq 2$. For each AE, a characteristic environment (CE) is built, an optimal robot based on the CE (RCE) is found, and $dF$ and $\alpha_{uf}$ are calculated. The proximity of the RCE and optimal robot based on the AE (RAE) is evaluated using two proximity indices: the percentage of AE for which $dF < 5\%$, depicted as $\gamma_{dF}$, and the percentage of AE in which the optimization constraint $\alpha_{uf} < \alpha_{allow}$ is fulfilled, depicted as $\gamma_{auf}$. The results of the comparison are presented in Fig. B.1. This graph shows, for example, that 90% of the CE based on $N_{mean} = 5$ provide $dF < 5\%$.

B.2 Example in Section 6.3.2

The characterization by the extreme targets was analyzed for the 100 random AE from the previous analysis. The dependence of $dL$, $dF$ and $\alpha_{uf}$ on $N_{ext}$ was evaluated by the indices $\gamma_{dF}$ and $\gamma_{auf}$. The values of the indices are presented in Table B.1 for the combinations of $N_{far}$ and $N_{near}$.

![Graph showing indices $\gamma_{dF}$ and $\gamma_{auf}$ as a function of $N_{mean}$ for 100 random environments.](image)

Fig. B.1 The indices $\gamma_{dF}$ and $\gamma_{auf}$ as a function of $N_{mean}$ for 100 random environments.
Table B.1  *Comparison of optimal robot based on an actual environment (RAE) and one based on a characteristic environment (RCE) with extreme characteristic targets for 100 random actual environments.*

<table>
<thead>
<tr>
<th></th>
<th>$N_{far}$</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_{near}$</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$\gamma_dF$ (%)</td>
<td>0</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>$\gamma_{auf}$ (%)</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

**B.3 Example in Section 6.3.2.6.3.3**

The combined characterization was analyzed for the 100 random AE from the previous analysis. The proximity of the RCE and the RAE was evaluated using the proximity indices $\gamma_dF$ and $\gamma_{auf}$. The values of the indices are presented in Table B.2.

Comparing the results of the analysis of the combined characterization with the analysis of the mean characterization, we can see that combined CE with a specific number of characteristic targets provide higher $\gamma_dF$ than the CE with the same number of mean characteristic targets, while $\gamma_{auf}$ is always 100%.

Table B.2  *Comparison of optimal robot based on an actual environment (RAE) and one based on a characteristic environment (RCE) with mean and extreme characteristic targets for 100 random actual environments.*

<table>
<thead>
<tr>
<th></th>
<th>$N_{mean}$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_{far}$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$N_{near}$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$\gamma_dF$ (%)</td>
<td>69</td>
<td>96</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>$\gamma_{auf}$ (%)</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>
**B.4 Example in Section 6.7**

**Example on Y-trellis (YT) apple tree model**

Views of the YT apple tree model are presented in Fig. B.2: tree model with obstacles (a), and side (b) and front (c) views with fruit presented as lines to distinguish their direction.

**RAE**

The RAE with different structures reaching the targets from the optimal locations are shown in Fig. B.3. Their optimal parameters are given in tables during the comparison with the RCE. The following robot structures are considered in this example: RRR (a), RRP (b) and PPP (c). The number of robot locations around the tree in these examples is $N_{loc} = 4$.

---

Fig. B.2  *Views of the Y-trellis apple tree model with obstacles (a). Side (b) and front (c) views with fruit presented as lines.*
Environment characterization

Three CE are built for the following maximal cluster sizes: $D_{\text{max,clust}} = 0.5$ m, $D_{\text{max,clust}} = 0.2$ m and $D_{\text{max,clust}} = 0.1$ m. These values of $D_{\text{max,clust}}$ correspond to 2, 4 and 6 clusters, respectively. The CE are shown in Fig. B.4, Fig. B.5 and Fig. B.6 in the frontal, side and isometric views.

Fig. B.4  CE for the YT apple tree actual environment with 2 clusters. The actual fruit are depicted as dots with different colors corresponding to different clusters. The mean targets of the clusters are depicted by asterisks with the cluster percentage. The farthest and nearest extreme fruit of the clusters are depicted as $\Delta$ and $V$, respectively. The assumed robot base position is depicted by a circle.
Fig. B.5  *Characteristic environments for the Y-trellis apple tree actual environment with 4 clusters. All signs are as in Fig. B.4.*

Fig. B.6  *Characteristic environments for the Y-trellis apple tree actual environment with 6 clusters. All signs are as in Fig. B.4.*

**Example on Central Leader (CL) apple tree model**

Views of the CL tree model are presented in Fig. B.7: tree model with obstacles (a), and side (b) and front (c) views with fruit presented as lines to distinguish their direction.
Fig. B.7 Views of the Central Leader apple tree model. The tree model with obstacles is shown in (a). Side (b) and front (c) views with fruit presented as lines.

RAE

The RAE with different structures reaching the targets from the optimal locations are shown in Fig. B.8. Their optimal parameters are given in tables during the comparison with the optimal RCE. The following robot structures are considered in this example: RRR (a), RRP (b), PPP (c). The number of robot locations around the tree in these examples is $N_{loc} = 4$.

Fig. B.8 Optimal robot based on an actual environment for Central Leader apple tree model with different structures: RRR (a), RRP (b), PPP (c).
Environment characterization

Three CE are built for the following maximal cluster sizes: $D_{\text{max,clust}} = 1$ m, $D_{\text{max,clust}} = 0.75$ m and $D_{\text{max,clust}} = 0.5$ m. These values of the $D_{\text{max,clust}}$ correspond to 3, 5, and 7 clusters, respectively. The CE are shown in Fig. B.9, Fig. B.10 and Fig. B.11 in the frontal, side and isometric views.

Fig. B.9  Characteristic environment for the Central Leader apple tree actual environment with 3 clusters. The actual fruit are depicted by dots with different colors corresponding to different clusters. The mean targets of the clusters are depicted as asterisks with the cluster percentage. The farthest and nearest extreme fruit of the clusters are depicted as Δ and V, respectively. The assumed robot base position is depicted by a circle.

Fig. B.10  Characteristic environment for the Central Leader apple tree actual environment with 5 clusters. All signs are as in Fig. B.9.

164
Fig. B.11 *Characteristic environment for the Central Leader apple tree actual environment with 7 clusters. All signs are as in Fig. B.9.*

**B.5 Example in Section 6.3.2**

Three robot structures (PPP, RRP and RRR) are checked for three tree models (Tall Spindle [TS], YT and CL). In this test, the robot with the optimal kinematics and optimal locations around the trees are taken. The performance of the robot is compared for the AE and an environment characterization which, contrary to the CE developed in Chapter 6, includes all of the original fruit along with reaching cones characterizing the branches and trellises. Hereafter we refer to this characterization as cone characteristic environment (CCE).

The result of the comparison is presented in Table B.3. The relative difference of the cost-function values for the robot working on the AE and CCE is depicted by $dF$. The percentage of unpicked fruit for the robot working on the CCE is depicted by $\alpha_{uf}$. The ratio between the time needed to solve the navigation problem for the AE, and the time needed to find the trajectory inside the reaching cones for the CCE is depicted as TR. Empty cells represent cases in which no solution fulfilling the optimization constraint was found.

According to the results of the comparison, the difference in the cost function can exceed 10% in some cases (marked by a gray background), up to 19%. The failure to solve the optimization problem with the CCE for the CL tree model in some cases is likely related to the large size of the environment. In such large environments, a robot located in a small number of locations, $N_{loc}$, finds complicated trajectories to reach far targets.
Table B.3 *Comparison of robot performance in the actual environment and the cone characteristic environment.*

| N_{loc} | | \(N_{loc}\) | | \(N_{loc}\) | \(N_{loc}\) | \(N_{loc}\) |
|---------| | PPP | RRP | RRR | PPP | RRP | RRR |
| TS | dF (%) | 4.8 | 13 | 5.2 | 8.5 | 0.1 | 11 | 8.8 | 4.2 | 7 | 14 | 19 | 11 |
| \(\alpha_{uf}\) (%) | 3.3 | 3.3 | 3.3 | 3.3 | 3.3 | 3.3 | 3.3 | 3.3 | 3.3 | 3.3 | 3.3 | 3.3 | 3.3 |
| TR | 2 | 3.1 | 4.8 | 5.6 | 2.6 | 3 | 5.3 | 6.3 | 2.8 | 3.5 | 4.5 | 6 |
| YT | dF (%) | 3.7 | 2.5 | 5.4 | 1 | 3.2 | 3.5 | 1 | 6.8 | 6.3 | 4.3 | 6.7 | 13 |
| \(\alpha_{uf}\) (%) | 5.9 | 4.4 | 7.4 | 4.4 | 8.9 | 5.9 | 4.4 | 4.4 | 5.9 | 4.4 | 4.4 | 4.4 |
| TR | 4.3 | 5 | 6.2 | 7.6 | 4.2 | 5.6 | 6.5 | 7.3 | 4.1 | 5.9 | 6.4 | 7.1 |
| CL | dF (%) | 7 | 2 | 2 | 7 | 2 | 8 | 11 | 7.9 | 7.2 | 3.6 |
| \(\alpha_{uf}\) (%) | 4.8 | 4.8 | 4.8 | 4.8 | 4.8 | 4.8 | 4.8 | 4.8 | 4.8 | 4.8 | 4.8 |
| TR | 8 | 9.4 | 7.1 | 9.6 | 6.5 | 7.6 | 8.3 | 10 | 166

These trajectories are far from the straight line in the robot joint space, and they are therefore poorly characterized by the reaching cones. The TS tree has large obstacle-free areas that cannot be entirely characterized by the reaching cones because the characterization is conservative. The actual trajectories for the robot motion are shorter than the trajectories based on the reaching cones, thus the cost-function values are smaller, and the difference \(dF\) is relatively large.

**B.6 Example in section 8.2**

The robot location optimization is compared for a number of tree models and the robot structures: CL apple tree, YT apple tree and YT apple tree row (YTR). The robot structures are PPP, RRP and RRR. The number of robot locations, \(N_{loc}\), is 2, 4, 6 and 8 for CL and YT, and 10 for YTR. The results of the comparison are given in Table B.4. The number of all possible locations and the number of suitable locations are written under the tree model name. The relative difference (in percent) between the robot performance cost function for the optimal robot location for all possible locations and the suitable locations is depicted by \(dF\). The sum of distances (in meters) between the corresponding locations is depicted as \(dR\). The total time ratio
between the two types of optimization is depicted as TR. Since the calculation for all values of $N_{loc}$ was performed in a single run, only one value of TR appears for each combination of tree model and robot structure. Empty cells represent cases in which no solution was found. Cells marked $\infty$ represent cases where the runtime of the solution for all possible robot locations is unacceptable.

Similar to the comparison for the navigation inside the reaching cones from the previous section, high values of $dF$ exceeding 10% (marked by gray) are related to the cases in which a large tree is operated by a robot located in a few locations. In most cases, $dF$ is below 5%.

Table B.4  **Comparison of the optimal robot location based on all possible robot locations and suitable robot locations.**

<table>
<thead>
<tr>
<th></th>
<th>PPP</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>$N_{loc}$</td>
<td>2</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>YT 92</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dF (%)</td>
<td>1.61</td>
<td>1.41</td>
<td>1.1</td>
</tr>
<tr>
<td>dR (m)</td>
<td>0.8</td>
<td>1.1</td>
<td>1.4</td>
</tr>
<tr>
<td>TR</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CL 114</td>
<td>38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dF (%)</td>
<td></td>
<td>1</td>
<td>0.27</td>
</tr>
<tr>
<td>dR (m)</td>
<td></td>
<td>1</td>
<td>0.82</td>
</tr>
<tr>
<td>TR</td>
<td>64</td>
<td></td>
<td></td>
</tr>
<tr>
<td>YTR 112</td>
<td>56</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dF (%)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dR (m)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TR</td>
<td>$\infty$</td>
<td>112</td>
<td></td>
</tr>
</tbody>
</table>

167
Appendix C. Statistical Analysis of Actual Environments

The optimal robot design is a time-consuming process, hence, we would like to reach some conclusions about the optimal robots and their working environment before solving the robot optimization problem. The main questions about the optimal robots are:

1. What number of modeled trees is sufficient to design an optimal robot for an entire orchard?
2. Do trees from the same orchard have similar geometry?
3. What are the limits of a robot's applicability? For example, if a robot is optimized for harvesting apples, what additional tasks is it able to perform?

If we already have optimal robots for given environments, these questions are easily answered. However, since we want to use the answers to simplify the robot design process, we can only answer these questions by analyzing the agricultural environment. For estimates of the sufficient number of trees to characterize an orchard and the similarity of the trees, we have to analyze the variability of the environment's geometrical features that affect the optimal robot.

D.1 Environment features

The robot's working environment consists of its targets and obstacles. In agricultural tasks, the plants represent the environment: the fruit are the robot’s targets, and the branches are its obstacles. The geometrical features of the fruit and branches define the robot's performance—its ability to reach the fruit while avoiding collision with the obstacles. The influence of environment geometry was discussed in Section 3.6.3. To evaluate this influence before designing the robot, the geometrical features of the environment are analyzed.

The fruit are characterized by their location. A fruit’s location has the following geometrical features: coordinates $X$, $Y$ and $Z$ of the fruit location, and distance $R$ to the $Z$ axis.
The branches are characterized by the reaching cones (defined in Chapter 7). The reaching cones have the following geometrical features: cone opening angle $\alpha_{open}$, angle of the cone axis relative to the tree row $\alpha_x$, angle of the cone axis relative to the ground $\beta$.

In this research, we consider the distribution of the plant features. As an example, we use the model of an apple tree Apple_30_10_13 from the model library, shown in Fig. C.1.

The histograms of the distributions of the fruit’s $X$ and $Z$ coordinates are presented in Fig. C.2.

![Model of an apple tree Apple_30_10_13 in three views.](image1)

**Fig. C.1** *Model of an apple tree Apple_30_10_13 in three views.*

![Histograms of the distribution of the $X$ (a) and $Z$ (b) coordinates of the fruit location.](image2)

**Fig. C.2** *Histograms of the distribution of the $X$ (a) and $Z$ (b) coordinates of the fruit location.*
D.2 Comparison of environments

The geometrical features of the environments are compared by their distributions. Two distributions are compared with the help of the Kolmogorov–Smirnov test. This test considers the shapes of the distributions as well as their boundaries. The test returns 0 if the distributions are equal. The value of the test is depicted here as KST.

For example, the features of the model Apple_30_10_13 are compared with the model of another apple tree Apple_11_10_13, presented in Fig. C.3.

Fig. C.3 Model of an apple tree Apple_11_10_13 in three views.

Fig. C.4 Histograms of the distribution of the X (a) and Z (b) coordinates of the fruit location for models Apple_30_10_13 (blue) and Apple_11_10_13 (green).
The distributions of the fruit’s $X$ and $Z$ coordinates for models Apple_11_10_13 and Apple_30_10_13 are compared and presented in Fig. C.4.

The Kolmogorov–Smirnov test returns $KST = 0.4$ for the comparison of the $X$ coordinate distributions, meaning that the distributions are relatively different, and $KST = 0.12$ for the comparison of the $Z$ coordinate distributions, meaning that the distributions are relatively close. The results of the comparison for all considered features are presented in Table C.1. The sum of the KST values is depicted by $\Sigma$. The values of the KST, indicating the strict difference or strict similarity of the distributions, cannot be defined because of high diversity of the tree features.

To investigate the similarity of the features, the model of a nectarine tree Nectarine_6_4_14, presented in Fig. C.5, is compared with the model of the apple tree Apple_30_10_13.

The results of the comparison of trees of different types are presented in the second row of Table C.1. For the most part, the differences between the trees of the same type (apples) are smaller than the differences between the trees of different types (apple and nectarine); nevertheless, a strict threshold cannot be defined.

Fig. C.5  *Model of a nectarine tree Nectarine_6_4_14 in three views.*

Table C.1  *Comparison of the geometrical features of two pairs of tree models by the Kolmogorov–Smirnov test.*

<table>
<thead>
<tr>
<th>KST, Feature</th>
<th>X</th>
<th>Z</th>
<th>R</th>
<th>$\alpha_{\text{open}}$</th>
<th>$\alpha_x$</th>
<th>$\beta$</th>
<th>$\Sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple_30_10_13-Apple_11_10_13</td>
<td>0.4</td>
<td>0.12</td>
<td>0.19</td>
<td>0.14</td>
<td>0.28</td>
<td>0.19</td>
<td>1.3</td>
</tr>
<tr>
<td>Apple_30_10_13-Nectarine_6_4_14</td>
<td>0.56</td>
<td>0.31</td>
<td>0.3</td>
<td>0.25</td>
<td>0.35</td>
<td>0.17</td>
<td>1.9</td>
</tr>
</tbody>
</table>
D.3 Feature similarity

The groups of trees from the same orchards (available in the model library) are compared and analyzed. Features of trees of different types and from different orchards are also analyzed to provide a reference for the comparison. The compared trees are divided into groups: each group includes trees of the same type (same orchard and same training method). The following groups of trees are compared:

1. Five apple trees trained by the Central Leader method (Apple 1).
2. Five nectarine trees shaped by the vase method (Nectarine 1).
3. Two apple trees shaped by the Spindle method (Apple 2).
4. Two nectarine trees shaped by the vase method (Nectarine 2).
5. Two peach trees shaped by the vase method (Peach).
6. One tangerine tree (Tangerine).

The results of the comparison are presented in Fig. C.6 in the following way. Each feature has a column consisting of two sub-columns: the left column (thick lines) represents KST values for a pair of plants of the same type (e.g., two trees of the Nectarine 1 group), and the right column (red thin lines) represents KST values for a pair of plants of different types. The comparison shows that there is no strict distinction in the geometrical features between trees of different types. The KST values depend on the features as well as on the tree types. For example, the distribution of the cone axis angle $\beta$ has mainly low values for the group Nectarine 1 and relatively high values for the group Apple 1. The values of the Nectarine 1 group are relatively low for the feature $\beta$, and relatively high for the feature $X$.

The only group that can be identified by its geometrical features is the group Nectarine 1 with the sum $\Sigma$ of the KST values as the most distinctive feature. The rest of the groups have high and low values of KST, showing that different groups are similar in some features and diverse in others.
Fig. C.6  *Comparison of tree groups by the Kolmogorov–Smirnov test.*

The comparison presented in Fig. C.6 cannot provide strict conclusions, although answers to questions 2 and 3, asked at the beginning of the appendix, can be approximated. Since some of the geometrical features are similar between different types of trees, the optimal robot for one type of tree may be able to perform tasks on the other type of tree, albeit not optimally. Some tree types have distinctive features, different from the other types, hence, their optimal robot is different from the optimal robots of the other types.

To answer question 1, some intuitive statistical considerations are used. A group of modeled trees is sufficient to characterize the entire orchard if the addition of an additional tree to the group does not provide additional significant information. We can evaluate the additional information by comparison of the distributions of the tree features: if the KST resulting from a comparison of the distribution of some feature of the additional tree and one of the trees in the existing group has a low value, the distribution of this feature in the additional tree is similar to its distribution in the existing tree. Hence, the additional tree does not add significant information.
about the distribution of this feature. If, according to this rule, for each compared feature there exist trees giving low values of KST when compared to the additional tree, the additional tree does not add information for any of the features, and thus it is redundant.

The described situation is likely impossible because of the high diversity of tree features. Nevertheless, we can try to evaluate the redundancy through the results of the comparison presented in Fig. C.6. We consider here only the groups Apple 1 (blue lines) and Nectarine 1 (green lines) consisting of five trees. We also assume that a low KST value is lower than 0.1, and a high value is higher than 0.3.

Consider the distribution of the fruit's Z coordinates for group Apple 1. All of the lines are above the value 0.1, meaning that all of the feature distributions are different; consequently, all of the trees in this group add additional information about the distribution of the fruit's Z coordinate. Thus, we have to check an additional tree to determine whether it can add information about the distribution of the fruit's Z coordinate.

In contrast, about half of the lines of group Nectarine 1 are close to the value 0.1. If, after adding an additional tree to the group, all five additional lines have high KST values, the additional tree obviously adds information about the distribution. If at least one line has a low KST value, the additional tree does not add information and is redundant for this feature.

Applying these considerations to the rest of the features, we can conclude that neither group Apple 1 nor group Nectarine 1 have a sufficient number of trees to characterize all of the features, and additional trees are required.
Appendix D. Initial Field Experiment

D.1 Description of the experiment

A field proof-of-concept experiment using the FFRobotics arm was conducted to demonstrate the methodology of the task-based design of the robotic arm kinematics. The goal of the experiment was to show how the performance of the robot is influenced by the robotic arm kinematics, and that the optimal kinematics found in the experiment corresponds to the optimal kinematics found by the optimization based on the tree model.

The experiment consisted of three parts, including fruit picking by a robot with three different kinematics of a robotic arm with changeable kinematics. The time of the robotic arm motion and the success rate were measured. To ensure identical environments for the comparison of robotic arms with different kinematics, we used the same tree with artificial fruit attached to the tree in the locations of the actual fruit from the previous season.

To simplify the experiment, the kinematics of the different robotic arms differed by the angle $\theta_2$, defining the angle of the third link relative to the Z axis of the previous link (according to DH notation), while the other kinematics parameters were held constant. Three angles were tested: $90^\circ$, $70^\circ$ and $60^\circ$.

The experiment was conducted during the winter season on an apple tree with no leaves or fruit. A single robot location was tested in the experiment. Eleven artificial fruit were attached to branches in the reachable area of the robotic arm. To provide controlled lighting conditions to simplify the fruit-recognition phase, the tree and the robot were covered by curtains. All of the fruit were recognized by the image-processing algorithm (developed by FFRobotics). The apple tree and the artificial fruit used in the experiment are presented in Fig. D.1.
The robot in the field experiment. The platform with the robotic arm (a). The robotic arm approaching one of the artificial fruit (b). The angle $\theta_2$ equals 70° in the figure.

D.2 Description of the robotic arm

The platform

The FFrobotics fresh-fruit robotic harvesting model consists of a platform and a robotic arm. The platform is a “Whooshh Innovations” mobile harvest platform vehicle (the “PT MHS”) (Fig. D.2a) that can move inside an orchard row, has picking stations for four workers with adjustable height and width, and utilizes the modular Whooshh fruit (vacuum) tubes for fruit transportation. The FFrobotics arm was installed on one of the picking stations.

Mechanical structure of the robotic arm

The FFrobotics arm is an experimental self-assembled arm with 3 DOF and a PPP structure. The arm actuators are Festo electrical actuators (toothed belt plain-bearing guide), representing the links of the robotic arm. The end effector is actuated by the Festo pneumatic rotary and linear piston. The actuators are driven by the Festo Stepper motor driver. The Whooshh fruit (vacuum) tube is attached to the second link of the robotic arm such that when the third link is in its nearest position, the end effector is located above the tube (Fig. D.1b).

End effector

The end effector is a simple soft underactuated three-finger gripper (Fig. D.2d). A single motor actuates all three fingers together up to the predefined force. After grasping the fruit, the end effector rotates to ensure safe fruit detachment.
Fig. D.2  The harvesting system. (a) The Whooshh platform with the vacuum tube for fruit transport and the picking stations for the workers. (b) The robotic arm installed on one of the picking stations of the platform (c) performs the apple harvesting inside an orchard. (d) The robot end effector is a soft gripper.

D.3 Comparison between simulation and experiment

The field experiment was conducted on the tree model in Fig. D.3 (winter, Golan Heights, Pink Lady, CL training system).
Fig. D.3  Actual task environment (a) and its model (b).

The results of the field experiment are detailed in Table D.1. The total robot motion time, the average robot motion time ($F$ defined in Section 4.1), and the number of picked fruit are different for different $\theta_2$ angles, with an optimal value of 70° providing the highest rate of picked fruit: 10 out of 11 (91%), despite the fact that the average picking time $F$ was the longest.

According to Table D.1, the lower the number of picked fruit, the lower the cost-function value $F$. The reason for this is as follows. All of the fruit nearest to the robots with no obstacles in between are picked in approximately the same time by all robots. Approaching the farthest fruit takes more time than approaching the nearest fruit, hence, the robots which are able to move through the obstacles to the farthest fruit have longer average performance time.
Table D.1  Performance of the experimental PPP robotic arm depending on θ₂.

<table>
<thead>
<tr>
<th>θ₂</th>
<th>Experiment</th>
<th>Simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>90°</td>
<td>70°</td>
</tr>
<tr>
<td>Total time (s)</td>
<td>23</td>
<td>33</td>
</tr>
<tr>
<td>F (s)</td>
<td>2.8</td>
<td>3.3</td>
</tr>
<tr>
<td>Picked fruit</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>Picked fruit (%)</td>
<td>73</td>
<td>91</td>
</tr>
<tr>
<td>α uf (%)</td>
<td>27</td>
<td>9</td>
</tr>
</tbody>
</table>

The optimal kinematics is defined by the constraint on the successful picking rate, $\alpha_{uf} < 15\%$. Since all of the fruit were recognized, the picking rate depends only on the presence of obstacles in the robot’s trajectory to the fruit. The robotic arm with $\theta_2 = 70^\circ$ had the smallest number of obstacles, providing $\alpha_{uf} = 9\%$.

The experimental robotic arm and its environment were modeled and simulated. The models of the robotic arm with different angle $\theta_2$, simulating the experiment, are shown in Fig. D.4. The results presented in Table D.1 based on the simulation corresponded to the results of the experiment. The number of picked fruit was the same for all cases. The robot average motion time $F$ differed between the experiment and the simulation because of the simplified modeling of the robotic arm dynamics.

The experiment showed that the kinematics of a robot influences the number of reachable fruit and the average motion time of the robot. Since there is a correspondence between robot performance in the AE and its simulation, the optimal kinematics of the robot can be found by the optimization based on the robot and environment models.
Fig. D.4  Fruit-picking experiment and its simulation. The actual robotic arm and the tree (a) are compared with their model (b). The models of the robotic arms with the different angles (90°, 80°, 70°, 60° and 50°) are presented in (c). The unpicked fruit are colored in black.
Bibliography


http://algorithmicbotany.org/ Website of the Biological Modeling and Visualization research group in the Department of Computer Science at the University of Calgary.

שיטות להכנת מכביי פירות עץ ומוצר רוובוט קוטף פירות

ויקטור בלוך
שיטות لت תוכן מקבלי של מסע עץ ורובוט קוטף

פירות

היבר על מחקר

לשם מילוי חלקי של הדרישות לקבלת תואר
דוקטור לפילוסופיה

ייקוטר בלור

הוגש לסנט הטכניון – מכניץ טכנולוגי לישראל
 חיפה
 יוני 2017
 סיוון 5777
המחק על בוגריית

פור opcion ימני דגני בפקולטה לאסכולות אדרחות
וסבים בנות ודר' אביטל בכר במכל ללהדסה חקלאית, ARO, מרכז וולקני

אני מודה למלגת נועמן ומלגת לבציון על התמיכה הכספית הנדרשת לתחזוקתי

אני מודה למלגת נועמן ומלגת לבציון

על התמיכה הכספית הנדרשת בהשתלמותי
הробוטים החקלאיים נחוקים כר צירם בשתי סכינים, או מעט רוטוטים הגיוני לשלב השימור המפורץ. לחות
הסובב השתמש הרובוטים המסחראים בשני החקלאים היכר הובנה של הרובוטים הקים אחר
 Canton מקשר שחקלאי לקרנוע ולתחזק את הנוה. על מתכת לחם, והרובוטים הזזים וירט ומכבשים
も多く ושClientes מ.GetHashCode להופמה. יד הצרור של הרובוט בتحمل Чающейoku תכנית
שורותיה הוא המסוגל לבצע את המשימה של של מתכת בכסף אופטימה שאמיתית עלמה,ยา
יושל והמשימה האמיתית בצורה, אולא הסביבה החקלאית היא מורכבת, בלתי מוכנה, ומיידור
כשיש ביתוא המידו של התאצאת mare ובתוליה את הרובוט. פתרון יעיל אופטימизация
ל킴טייה של הרובוט על קלח זמם וישב אל סבר בלולא מרכזת סברת העדשה.

במהלך זה פוחתה התתודוזיה לפיש של🌸 רבבה העבודה של הרובוט. פתרון יעיל אופטימציה
באמצעות המודול המפותח קור לפתור העביה, מושג בצומח שיבשו קר. התתודוזיה פותחה
באמצעת הבאים:
1. מודול הסביבה החקלאית בודרדו מקרית מדריך מכן:

המייגל התבגרת מ으면ים המ𝜂ורות המסחראים ברך בשתי סכינים מדויק. שירות את השימור
בצי מודול שיאצר מתמונות. שירות שיניים ישמה במודל אוזר בודרדו שיאצר מודל
לבני מודול של צים (L-systems). המודול של הצים כללים: עלמין, פורת, זרעי, וצומח
מתוך צומח. בناقשת ספירה של 20 צים, שטח צים שיאצר צים מאמץ מתכוע דומששה
עוצמ כל צים, שיאצר צים צומח צור מתמקמת שיווק
2. אופטימיזציה קינטייה של צור רוטוטים המיועד袢ת השימור המפורץ.

פונקציות המידי ביבי אופטימיזציה זה הוא זפון בקציפה הפור, תאי של אופטימיזציה
יהיו ציס מת הרובוט בOfYear מ-95 מפריש. משטחי האופטימיזציה הם חקל מפרשתות בפתול
השא או מיצוג את הקינטייה של הרובוט. בדלתצוף את הפרתון ב鸪צן המאחר
נבחנה שטח צים יושב. על מתכת לוחש את העפר של פונקציות המידי של רובוט, יש
ל fod את העברת הבאות: קינטייה הפוכה, יש, בדיקת התגשמות עם התקומם ומיקום
אופטימלי של מספר רוטוטים מסביר
3. אופטימיזציה מקומ של הזרוע הרובוטית.

בנтик עמדות אופטימלי של צור נקודת Förnerת בובר סיבוב העברת ייו צים שה
מצאת במקומ קובץ במרובע הש工業י של. ביסבוח חקליאיר הזורע לא צולח ליה ליישוב בקועה
תקציר

הרובוטים החקלאים נחקרים כבר עשרות שנים, אך מעט הרובוטים הגיעו לשלב השימוש המסחרי.

אחת הסיבות למיעוט הרובוטים המסחריים בענף החקלאות היא Цена המחירים של הרובוטים הקיימים אשר
אינו מאפשר לחקלאים לקנות ולתחזק אותם. על מנת להוריד את מחירים של הרובוטים
אנחנו מציעים לתכנן הרובוטים המיועדים למשימה הספציפית בפי במרכז.

שחורובית היא המסוגל לבצע את המשימה של של. על מתכת בכסף אופטימה שאמיתית עלמה,ยา
יושל והמשימה האמיתית בצורה, אולא הסביבה החקלאית היא מורכבת, בלתי מוכנה, ומיידור
כשיש ביתוא המידו של התאצאת mare ובתוליה את הרובוט. פתרון יעיל אופטימציה
ל킴טייה של הרובוט על קלח זמם וישב אל סבר בלולא מרכזת סברת העדשה.

במהלך זה פוחתה התתודוזיה לפיש של🌸 רבבה העבודה של הרובוט. פתרון יעיל אופטימציה
באמצעות המודול המפותח קור לפתור העביה, מושג בצומח שיבשו קר. התתודוזיה פותחה
באמצעת הבאים:
1. מודול הסביבה החקלאית בודרדו מקרית מדריך מכן:

המייגל התבגרת מ으면ים המ𝜂ורות המסחראים ברך בשתי סכינים מדויק. שירות את השימור
בצי מודול שיאצר מתמונות. שירות שיניים ישמה במודל אוזר בודרדו שיאצר מודל
לבני מודול של צים (L-systems). המודול של הצים כללים: עלמין, פורת, זרעי, וצומח
מתוך צומח. בناقשת ספירה של 20 צים, שטח צים שיאצר צים מאמץ מתכוע דומששה
עוצמ כל צים, שיאצר צים צומח צור מתמקמת שיווק
2. אופטימיזציה קינטייה של צור רוטוטים המיועד袢ת השימור המפורץ.

פונקציות המידי ביבי אופטימיזציה זה הוא זפון בקציפה הפור, תאי של אופטימיזציה
יהיו ציס מת הרובוט בOfYear מ-95 מפריש. משטחי האופטימיזציה הם חקל מפרשתות בפתול
השא או מיצוג את הקינטייה של הרובוט. בדלתצוף את הפרתון ב鸪צן המאחר
נבחנה שטח צים יושב. על מתכת לוחש את העפר של פונקציות המידי של רובוט, יש
ל fod את העברת הבאות: קינטייה הפוכה, יש, בדיקת התגשמות עם התקומם ומיקום
אופטימלי של מספר רוטוטים מסביר
3. אופטימיזציה מקומ של הזרוע הרובוטית.

בנтик עמדות אופטימלי של צור נקודת Förnerת בובר סיבוב העברת ייו צים שה
מצאת במקומ קובץ במרובע השانون של. ביסבוח חקליאיר הזורע לא צולח ליה ליישוב בקועה
בגלל הממדיםの大ים של שורות הצמחים ישörן פתatorio את בעיית המיקום האופטימלי של הדח作った הרובוטי מסביב לצמחים או לארק שורות הצמחים וז ويمكن במקלט למסڕב של משטחים. במכהק הזו פותחה שיטה המבוססת על הגהות של שיבת העבורה שמאפשרת.

לפתור אתSibling את השיבו טסיבר שובה של_boיט.

4. פיתוח שיטה לאופיין סיביבת העבורה של_boיט.

שיטה זו מיועדת לפישוט סיבוב של_boיט מסביב לשכונת המשרה המגזר של שיבת. שעירה

השיטה היא מציאת עבורה זמימה והולחמת בפעמת מאמין. ידי,ेस, כיוון שהעבורה ברווח מחוץ הבגר

מודחפת בבר阱://++; מאועט ייז שימון ארבעה הקבוצות שמקושרות המחייה של הרבר

היה קורבהלהלשה המחייה של כל הסיבוב חלקית והווחאה היו לשיבית אופיין שלleckות את כל התכונת הגיאומטרית שלremoveClass_index:K

אותופ היורים. של ראשון לשבונייה ייוולד את הסיבוב השכונת שמשמשת על הקידמית

האבטתרי:ל של_boיט. תכונת המגזר דלי. שלו זזונת הזרות למעורר סיבוב

המקושה. למחר BROK לששתתנ את ברכות שלימש מחקר זה:

א. חלקה פורט לקבוצת לאפקיו תמורות שאפשר לשיבית לשכונת הבקדקה על ידי

K-means clustering

ב. קונטס הגעה לאמקד el hakkah. דרך קוסמי הגהה המופה סלט-מדידי

שمالפיים את האזורים הנקשים במקשולים שבם ברכות יולי לוב. בנסך שים

בקהונס הגעה מקסשת את החושי של פנסים אגניטוב מתום ת춘 בעית

אבטתרי. זה מושע צורר במדיקה התהנהנויות בט נזרות בין המכשולים גתונז

בعباد הוזון.

5. גיוון התכונות גיאומטריות של סיבוב העבורה של_boיט.

התקנות הגיאומטריות של הסיבוב שпечתם במחק המקלט משמשות על התכונת הקידמית של

הזרות ש 무엇עד לעבד סיבוב הזרות. הגהות של התכונת יית להסרה מקסמת לע צורב הרובוט

לפי פורט ביני האבטתרי:ז מאפרמק לפסחי את המתיישן והלוכה להבצע מראה את

המיושם אליהם הרובוט דרש להנני רגל קורב מאמדם של הרובוט. קומס הגהה

קובעים את המיושם האבטתרי:ז של הרובוט ושל מפרח הרובוטי תדוש כידוע בᤊ את

ההפישת.

הכינו הנוסח במחק הוא הרחב התכן האבטתרי:ז של רובס לטן האופטימלי של המרחב

שוכרב מחזרות ומיסיבת העבורה של. במחק להנהלת המחק, הกายוטעה של רובס

משמשת מחזאות אבטתרי:ז הלוגס של הסיבוב, לכל FileNotFoundException של המבנה הסיבוב יולי לוחבי.
לרוב, אופטימלי טוב יותר מהרובוטים האופטימליים לסביבות לא מתוכננות. אבל, בניגוד לסביבת העבודה התעשייתית שקל להיעטף, הסביבה החקלאית מורכבת יותר.

מעצעים множטיים והתקן שלמה موقف על ידי כלים אגרונומיים.

ההליך הביאים של המחקר עוקפים בתוכנוubi:

1. בדיקת יעילות של שביל לצורת העצים.

במחקרים אגרונומיים נבדקה יעילות בתנובה ובטיפול של שיטות ושימוש לצורת העצים. במחקרים זה נועשה בדיקת יעילות של צורת העצים בעיצובים שונים ברובוטי. הרובוטים האופטימליים נמצאו למודלים של עצי תפוח בעיצובים שונים בקטיף ר. הרובוטים האופטימליים נמצאו למודלים של עצי תפוח בעיצובים שונים בקטיף ר. הרובוטים האופטימליים נמצאו למודלים של עצי תפוח בעיצובים שונים בקטיף ר. הרובוטים האופטימליים נמצאו למודלים של עצי תפוח בעיצובים שונים בקטיף ר. הרובוטים האופטימליים נמצאו למודלים של עצי תפוח בעיצובים שונים בקטיף ר. הרובוטים האופטימליים נמצאו למודלים של עצי תפוח בעיצובים שונים בקטיף ר. הרובוטים האופטימליים נמצאו למודלים של עצי תפוח בעיצובים שונים בקטיף ר. הרובוטים האופטימליים נמצאו למודלים של עצי תפוח בעיצובים שונים בקטיף ר. הרובוטים האופטימליים נמצאו למודלים של עצי תפוח בעיצובים שונים בקטיף ר. הרובוטים האופטימליים נמצאו למודלים של עצי תפוח בעיצובים שונים בקטיף ר. הרובוטים האופטימליים נמצאו למודלים של עצי תפוח בעיצובים שונים בקטיף ר. הרובוטים האופטימליים נמצאו למודלים של עצי תפוח בעיצובים שונים בקטיף ר. הרובוטים האופטימליים נמצאו למודלים של עצי תפוח בעיצובים שונים בקטיף ר. הרובוטים האופטימליים נמצאו למודלים של עצי תפוח בעיצובים שונים בקטיף ר. הרובוטים האופטימליים נמצאו למודלים של עצי תפוח בעיצובים שונים בקטיף ר. הרובוטים האופטימליים נמצאו למודלים של עצי תפוח בעיצובים שונים בקטיף ר. הרובוטים האופטימליים נמצאו למודלים של עצי תפוח בעיצובים שונים בקטיף ר. הרובוטים האופטימליים נמצאו למודלים של עצי תפוח בעיצובים שונים בקטיף ר. הרובוטים האופטימליים נמצאו למודלים של עצי תפוח בעיצובים שונים בקטיף ר. הרובוטים האופטימליים נמצאו למודלים של עצי תפוח בעיצובים השונים בקטיף ר. הרובוטים האופטימליים נמצאו למודלים של עצי תפוח בעיצובים שונים בקטיף ר. הרובוטים האופטימליים נמצאו למודלים של עצי תפוח בעיצובים שונים בקטיף ר. הרובוטים האופטימליים נמצאו למודלים של עצי תפוח בעיצובים שונים בקטיף ר. הרובוטים האופטימליים נמצאו למודלים של עצי תפוח בעיצובים שונים בקטיף ר. הרובוטים האופטימליים נמצאו למודלים של עצי תפוח בעיצובים שונים בקטיף ר. הרובוטים האופטימליים נמצאו למודלים של עצי תפוח בעיצובים שונים בקטיף ר. הרובוטים האופטימליים נמצאו למודלים של עצי תפוח בעיצובים שונים בקטיף ר. הרובוטים האופטימליים נמצאו למודלים של ע pii התוכנה תכנית בקטיף ר. הרובוטים האופטימליים נמצאו למודלים של ע pii התוכנה תכנית בקטיף ר. הרובוטים האופטימליים נמצאו למודלים של ע piiהתוכנה תכנית בקטיף ר. הרובוטים האופטימליים נמצאו למודלים של ע piiהתוכנה תכנית בקטיף ר. הרובוטים האופטימליים נמצאו למודלים של ע piiהתוכנה תכנית בקטיף ר. הרובוטים האופטימליים נמצאו למודלים של ע piiהתוכנה תכנית בקטיף ר. הרובוטים האופטימליים נמצאו למודלים של ע piiהתוכנה תכנית בקטיף ר. הרובוטים האופטימליים נמצאו למודליות של ע piiהתוכנה תכנית בקטיף ר. הרובוטים האופטימליים נמצאו למודליות של ע piiהתוכנה תכנית בקטיף ר. הרובוטים האופטימליים נמצאו למודליות של ע piiהתוכנה תכנית בקטיף ר. הרובוטים האופטימליים נמצאו למודליות של ע piiהתוכנה תכנית בקטיף ר. הרובוטים האופטימליים נמצאו למודליות של ע piiהתוכנה תכנית בקטיף ר. הרובוטים האופטימליים נמצאו למודליות של ע piiהתוכנה תכנית בקטיף ר. הרובוטים האופטימליים נמצאו למודליות של ע piiהתוכנה תכנית בקטיף ר. הרובוטים האופטימליים נמצאו למודליות של ע piiהתוכנה תכנית בקטיף ר. הרובוטים האופטימליים נמצאו למודליות של ע piiהתוכנה תכנית בקטיף ר. הרובוטים האופטימליים נמצאו למודליות של ע piiהתוכנה תכנית בקטיף ר. הרובוטים האופטימליים נמצאו למודליות של ע piiהתוכנה תכנית בקטיף ר. הרובוטים האופטימליים נמצאו למודליות של ע piiהתוכנה תכנית בקטיף ר. הרובוטים האופטימליים נמצאו למודליות של ע piiהתוכנה תכנית בקטיף ר. הרובוטים האופטימליים נמצאו L

2. תכנון עצים כסביבת ה-active של רובוט.


1. בדיקת יעילות של שביל לצורת העצים.

במחקרים אגרונומיים נבדקה יעילות בתנובה ובטיפול של שיטות ושימוש לצורת העצים. במחקרים זה נועשה בדיקת יעילות של צורת העצים בעיצובים שונים בקטיף ר. הרובוטים האופטימליים נמצאו L

2. תכנון עצים כסביבת ה-active של רובוט.

בדייקת שיטות העיצוב הקיימות יכלול הת 개인정보 על השיטה הקיימת האופטימלית. לצורך סNSNotification, תכנון הפנימי של השיטה היא תכנית אופטימלי, alcuni פיתוחים שונים ומורכבים בשיטת העיצוב H-trellis של עצי תפוח. בא kenniktor של צורת העיצוב Y-trellis המודרנט משמשת עצי תפוח. בא kenniktor של צורת העיצוב Y-trellis המודרנט משמשת עצי תפוח. בא kenniktor של צורת העיצוב Y-trellis המודרנט משמשת עצי תפוח. בא kenniktor של צורת העיצוב Y-trellis המודרנט משמשת עצי תפוח. בא kenniktor של צורת الع

Seleccionamos como resultados de estas investigaciones un modelo de diseño de árboles que utiliza diferentes técnicas de diseño y se ajusta a diferentes tipos de árboles. Estas técnicas de diseño pueden ser optimizadas para diferentes tipos de árboles y condiciones ambientales. En este estudio, se evaluaron diferentes técnicas de diseño de árboles usando diferentes métodos y se compararon varias técnicas de diseño de árboles Para determinar la eficiencia de diferentes técnicas de diseño de árboles. Cada una de estas técnicas de diseño se evaluó utilizando diferentes métodos y se compararon varias técnicas de diseño de árboles para determinar la eficiencia de diferentes técnicas de diseño de árboles. Penturas Atractor,