BEN-GURION UNIVERSITY OF THE NEGEV
FACULTY OF ENGINEERING SCIENCES
DEPARTMENT OF INDUSTRIAL ENGINEERING AND MANAGEMENT

Learning motion primitive parameters for a Medjool date thinning robot

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE M.Sc DEGREE

By: Or Bar-Shira

September 2019

BEN-GURION UNIVERSITY OF THE NEGEV
FACULTY OF ENGINEERING SCIENCES
DEPARTMENT OF INDUSTRIAL ENGINEERING AND MANAGEMENT

Learning motion primitive parameters for a Medjool date thinning robot

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE M.Sc DEGREE

By: Or Bar-Shira

Supervised by: Prof. Sigal Berman

Author: Or Bar-Shira Or .... Date: 28/09/2019

Supervisor: Sigal Berman [illegible] .... Date: 28/09/2019

Chairman of Graduate Studies Committee: [illegible] .... Date: 28/9/2019

September 2019

Abstract

Fruitlets thinning is an important task of Medjool date cultivation. Efficient and timely fruitlets thinning is essential for attaining high quality yields. Currently, fruitlets thinning requires precision, is time critical, and is labor intensive. The estimated time required to thin a single mature tree is 3.5 hours and more than a million hours of labor are currently required to thin all the Medjool trees in Israel. Automation of thinning is essential to enable fruitlets thinning of larger plots using reduced manpower during the thinning season.

Motion control during thinning is challenging as careful into-the-canopy motion is required. Furthermore, the environment is dynamic, therefore run time trajectory adaptation is also essential. We present a development of a motion planning and control method for a robotic system for Medjool date thinning. The method is based on dynamic movement primitives (DMPs). DMPs are a set of nonlinear equations used for encoding motion policies, based on dynamical systems which have parameters whose values can be tuned to facilitate performance of new tasks. The DMP parameters determine the precision of the generated movement and their values are influenced by the task, the robot, and the environment. DMP parameters can be divided into three categories: shape parameters (defining motion trajectory), meta parameters (determining the DMPs frame of operation, e.g., goal point), and external parameters (determining external, high level, environmental constraints that affect DMP operation, e.g., motion start time).

We present a framework for predictable DMP parameter adaptation, based on forming a mapping from parameters representing task requirements to meta parameters. The mapping facilitates generalization and precision when generating a new movement, both are essential qualities for thinning. Furthermore, the mapping is formed a-priori, hence it ensures predictable and robust task performance during run time. Since the mapping is a multi-input-multi-output learning problem, it can be modeled using a deep neural network, in which the task parameters form the input layer and the meta parameters are the output layer. Network training was expedited by analyzing the parameter space using principal component analysis and locally linear embedding.

To attain good accuracy, the deep neural network requires a large dataset. Lack of agricultural annotated datasets is considered among the main reasons for lower performance of advanced machine learning algorithms in agriculture and among the main bottlenecks for introducing robotics in agriculture. As establishing a dataset of physical measurements is challenging, a method for constructing annotated dataset that is based on artificial data was developed. A validated, stochastic model of a Medjool date fruit bunch was created and visualized in 3D using python OpenGL. The fruit bunch is modeled as an assembly of geometric shapes, each shape has a unique distribution fitted to it.

The ability to generate an accurate movement utilizing the adapted DMP method was tested in a preliminary experiment in a simulation of a ball throwing task, where the task parameter is the ball's landing point on the floor. Two simulation models, ballistic and adapted, are presented. Furthermore, the deployment of the deep neural network is compared with that of kernel regression. The parameter space analysis indicated that the complexity of both, ballistic and adapted, models was moderate with stronger constrains in the adapted model. The results showed that for large datasets both mapping methods attained low movement error (beneath 12% distance and 2° angle). A second experiment examined the competence of the deep neural network to learn the task and meta parameter mapping for the Medjool thinning task based on datasets generated from the fruit bunch model. The parameter space analysis indicated that the problem complexity was high. After training, the network achieved very low error of 2.6%. The results are promising, and the method and tools developed are ready for an upcoming field test.

Keywords: Dynamic movement primitives (DMP), robotics in agriculture, deep neural networks, artificial data.

Acknowledgements

I would like to express my deep gratitude to Professor Sigal Berman, my supervisor, for her patient guidance, enthusiastic encouragement and useful critique. Her experience, professionalism, and willingness to help assisted me in guidance and in finding the best methods to perform this study; I take pride in acknowledging the experts who were involved in the forming and validation of the fruit bunch model for this research project: Prof. Avital Bechar, Dr. Yuval Cohen, and Mr. Avraham Sadowsky. Without their participation and input, the building and validation of the model could not have been successfully conducted; I would also wish to thank my lab colleagues, Mr. Yosef Cohen and Mrs. Tal Shushan for their wonderful collaboration.

This research is supported by the Israeli ministry of agriculture and the Israeli Date Grower's board in The Plant Council.

Table of Contents

List of tables

List of figures

List of abbreviations and nomenclature

AI	Artificial Intelligence
AS [°]	Spikelets minimal bounding box angle
DMP	Dynamic Movement Primitives
DNN	Deep Neural Networks
FL	Fruitlets Load
H [cm]	Gripper end point height
HBM	Hierarchical Bayesian Models
L [cm]	Spikelets remaining length
LLE	Locally Linear Embedding
LS [cm]	Spikelets minimal bounding box length
MLP	Multilayer Perceptron
PA	Precision Agriculture
PC	Principal Component
PCA	Principal Component Analysis
r [cm]	Ball landing radius
R [cm]	Gripper end point radius
RL	Reinforcement Learning
TDMP	Tight Dynamic Movement Primitives
WS [cm]	Spikelets minimal bounding box width
x	x coordinate of a point
y	y coordinate of a point
z	z coordinate of a point
α [°]	Ball landing horizontal angle
β [°]	Gripper end point horizontal angle
τ [sec]	Motion duration

Chapter 1: Introduction

1.1 Background

Precision agriculture (PA) is an accurate agriculture that is based on a decision support system that take into consideration observations, measurements, and statistical analysis in order to optimize cost and value (Zhang, Wang, & Wang, 2002). The interest in application of autonomous robotics to precision agriculture is growing notably due to the need to keep up with the increasing demand for productivity and quality of food production whilst decreasing the resources required, reducing chemical treatments, and promoting sustainability (Bac, et al., 2014; Potena, Nardi, & Pretto, 2016). Yet, the introduction of robotics in agriculture is challenging. This is mainly due to the unstructured and dynamic environment that impair the robot's ability to adapt to changes, a mandatory requirement for integration of robots in agricultural enviroments (Arkin, 1998; Zhou & Zhang, 2019). Another elment considered among the main bottlenecks for introducing robotics in agriculture is the performance of artificial intelligence (AI) algorithms. In recent years grate efforts are dedicated to adding a degree of AI to the robot operating mechanism and thus enable increased selectivity, precision, and robustness. However, there is a great absence of detailed and large annotated agricultural datasets that current state-of-the-art AI methods require, data that is infeasible to obtain manually (Barth, et al., 2018).

1.2 Objectives and contribution

This research contributes to the combination of the worlds of AI and robots in agriculture. The research is conducted in the domain of Medjool date fruitlets thinning. Manual methods for fruitlets thinning that are common nowadays are not here to stay. Other methods such as autonomous or semi-autonomous systems can be much more efficient. Previous studies implemented precision agriculture methods to improve the process of growing dates. Their focus varied from the level of a single tree, to a grove, and more widely on a regional level (Cohen, 2014). Yet, no work was done on the use of precision agriculture for fruitlets thinning of dates and more specifically on automation of fruitlets thinning with robots. Automated thinning can be conducted by shortening the fruit bunch. Both, the deciding where to cut and the appropriate cut without damaging other plant parts are complex. Advanced decision making is required, along with inside-the-canopy motion which entails a dynamic control accounting for interaction with the environment.

Our first objective was to develop a movement algorithm for a robotic arm, suited for dynamic environment, that relies on a deep neural network for decision making. Our second objective was to develop a method for generating artificial datasets to train the deep neural network needed for motion planning thus enabling the generation of the movement albite the absence of real data.

A 3-dimensional model of a Medjool date fruit bunch and its visualization were built and were validated by two physiologists, experts in Medjool date cultivation. Two experiments with two different tasks were conducted: robotic soft ball throwing and Medjool thinning. The robotic soft ball experiment tested the feasibility of the proposed movement algorithm and compared the performance of the deep neural with that of kernel regression. The Medjool thinning experiment examined the ability of the network to train on the artificial data generated from the fruit bunch model and to perform new tasks.

Two papers were submitted and are pending response. The first paper was submitted to IEEE Robotics and Automation Letters (RA-L) and ICRA (Q1), the second was submitted to Robotics and Autonomous Systems (Q2). Preliminary work was published in abstracts presented in the 21st National Conference on Industrial Engineering and Management, XXXVIII CIOSTA & CIGR V International Conference, and the 6th Israeli Conference on Robotics.

1.3 Research scope and limitations

This work is part of multi-disciplinary project that is funded by the Israeli ministry of agriculture. A prototype consisting a thinning tool and a sensing system that are placed on a robotic arm is developed. The sensing system comprises a camera and a sonar sensor. The prototype will be mounted on an altitude tool, the robotic arm will approach a fruit bunch, the perception apparatus will estimate the properties of the fruit bunch, based on this a decision-making process will determine the level of thinning required, and the mechanical thinning system will perform the thinning.

The mission of the Ben-Gurion group is to develop and adapt algorithms for planning the movement of the robotic arm in space to control the sensing and thinning. the current study presents the movement algorithm developed for the robotic arm which is based on dynamic movement primitives (DMPs) and on adaptation of the DMP goal during run time using a mapping

based on a deep neural network. Furthermore, it presents a development of a stochastic model of the Medjool's fruit bunch for generating datasets that enable training the network.
This work does not include the processing of the sonar and image input. The image processing was performed by Mrs. Tal Shushan, a M.Sc. student in the Department of Industrial Engineering and Management at Ben-Gurion University of the Negev under the supervision of Prof. Sigal Berman. The final algorithm was not field tested yet as the prototype is not yet completed. Furthermore, the prototype needs to be tested in a limited period during thinning season (the upcoming 2020 thinning season is in April 2020).

In the robotic soft ball throwing experiment, the current study presents the use of a deep neural network to learn a mapping between task and meta parameters. The two simulation models that were built for generating the datasets for this experiment and the use of kernel regression to learn a mapping were performed by Mr. Yosef Cohen, a Ph.D. student in the Department of Industrial Engineering and Management at Ben-Gurion University of the Negev under the supervision of Prof. Sigal Berman.

1.4 Outline

This thesis is arranged as follows: Chapter 2 provides a literature review on the method used to create the robot control policy and learning algorithm, prior studies, and essential concepts related to the work. Chapter 3 describes a mathematical model of a fruit bunch that was devised and its utilization for generating annotated datasets. Chapter 4 details the methodology for learning motion parameters developed for this thesis. Chapter 5 describes two experiments held, chapter 6 describes the results of the experiments, discussion and future work are provided in chapter 7.

Chapter 2: Literature review

2.1 Overview

The current chapter reviews the concepts and methods relevant for developing a dynamic movement algorithm for a robotic arm for fruitlets thinning of Medjool date. For promoting understanding of the required automation section 2.2 describes Medjool dates, the Israeli Medjool market, and the importance and difficulties of the fruitlets thinning operation. Since this work deals with adaptation of dynamic movement primitives (DMPs) section 2.3 reviews movement primitive representations in robotics. In addition, it describes methods for improving the ability of DMP adaptation to new tasks using parameters. The main contribution of this work is in the use of deep neural networks (DNNs) in an agricultural environment and in the context of tuning DMP parameters therefore section 2.4 reviews DNNs, the difficulty in establishing training databases, and specifically the use of DNNs for adapting motion parameters.

2.2 Fruitlets thinning of Medjool dates

2.2.1 Medjool dates

Medjool dates are considered premium dates (Bar-Shira, et al., 2019). Originated from Morocco, the cultivar was first introduced to Israel at the 1950s and mostly during the early 1970s (Cohen & Glasner, 2015). The medjool dates grow upon fruit bunches (Figure 1). The Medjool date fruit bunches grow in whorls within the date crown that reaches tens of meters in height (Chao & Krueger, 2007). Farmers usually allocate the fruit bunches to three typical whorls (Top, Middle, Bottom) according to their location in the crown. Each fruit bunch is made up of a fruit bunch rachis, spikelets, and fruitlets (Figure 2). The rachis grows in an upward direction, titled with respect to the palm trunk. During the season, as the fruitlets become larger, the tilt increases due to their weight. The spikelets grow in a helical arrangement from the top part of the rachis, they are allocated by farmers to clusters (similar to whorls for fruit bunches). The fruitlets (small fruits) develop from flowers along the spikelets in a process termed fruit-set (Cohen & Glasner, 2015; Bar-Shira, et al., 2019).

Medjool dates cultivation can be divided into six main periods (Figure 3). The annual cycle starts with the removal of spines from bases of all leaves developed during the previous year. During flowering, between February and April, farmers collect pollen from male trees and pollinate female flowers, leading to the process of fruit-set. Fruitlets thinning is performed in April and May. At the end of this period, fruit bunches are supported on leaves, fastened, and covered with bags. During August–October, fruit harvesting takes place (Cohen & Glasner, 2015).

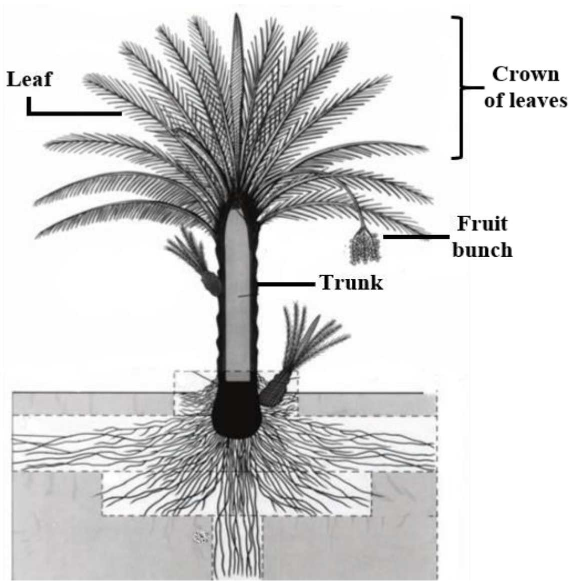


Figure 1: Schematic diagram of the date palm (adapted from Chao & Krueger, 2007).

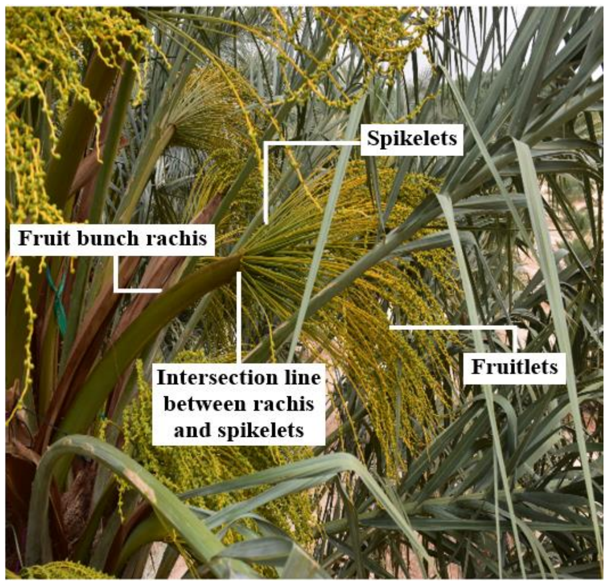


Figure 2: Medjool date fruit bunch (early thinning period) (Bar-Shira, et al., 2019).

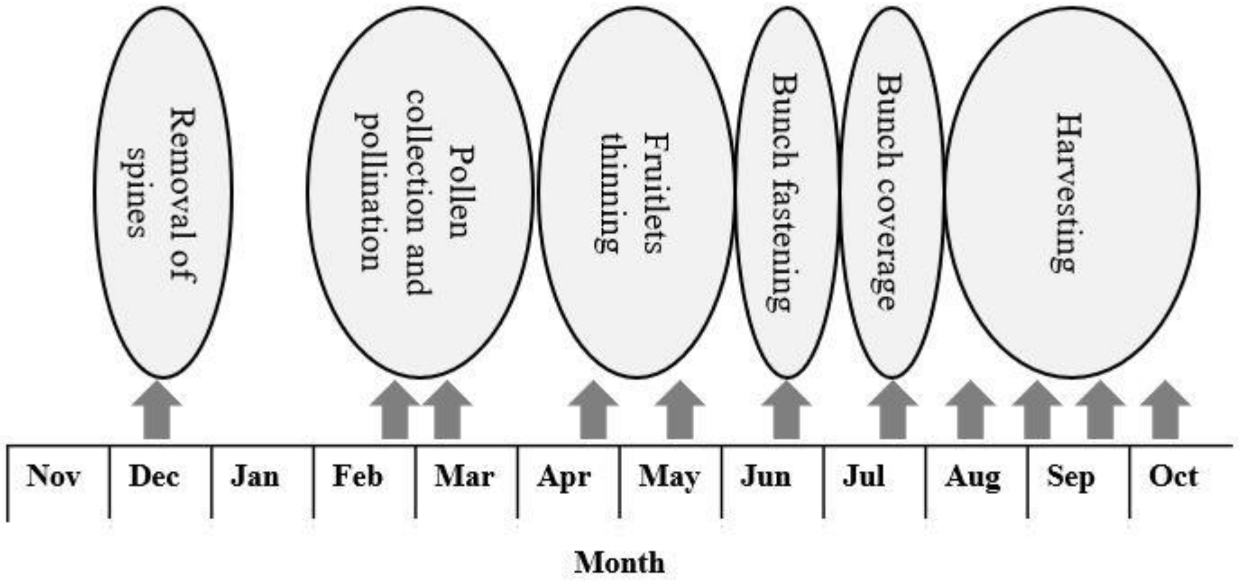


Figure 3: Schematic working calendar for Medjool orchards in southern Israel. Arrowheads represent tasks enquiring physical climbing to the crown of the tree (Cohen & Glasner, 2015).

2.2.2 The importance of Medjool dates to the Israeli market

Dates are the main plantation crops in the arid regions of Israel, from the Sea of Galilee, along the Jordan Valley to the southern Arava. In the southern parts of Israel, the date industry is the major and sometimes the only horticulture industry and its importance to the economy of these regions is immense (Cohen & Glasner, 2015). The Medjool is the main date-palm cultivar in Israel, where about 75% of all planted trees are Medjool (Hausler, Fridkin, & Kachel, 2017). In addition to their importance for the livelihood of the residents of southern Israel, Medjool dates are of great value to the Israeli economy. In the past decade dates have become one of the highest value products that Israel export, most of them are exported to the European markets. In 2015, the volume of exported dates was estimated at 44% of the dates production and the export value of the industry was estimated at 400 million NIS. The Israeli Medjool has a market share of between 65% to 75% of the amount of Medjool exported worldwide, positioning Israel as one of the major exporters of Medjool in the international markets (Hausler, Fridkin, & Kachel, 2017). The main limitation slowing down market expansion is that Mejdool date production is labor intensive. The most manual intensive task in the production cycle is thinning (Bar-Shira, et al., 2019).

2.2.3 Thinning

Thinning is a term used in agriculture which mean the removal of some parts of the plant in order to make it less dense and obtain better growth of the parts that remain (Lewis, 2001). It has been shown that fruitlets thinning of date palms lead to an improvement of the yield and fruit quality (Moustafa, 1998). Yet, it is considered the most labor intensive period in the cultivation of Medjool dates since it varies from one palm tree to another and it is performed manually. The optimal thinning requires consecutively removing two out of three fruitlets along each spikelet, enabling the development of 1/3 of the fruitlets equally spread upon the spikelets. This procedure is not performed in Israel since it is extremely time consuming (Cohen & Glasner, 2015). In practice, manual thinning is performed by removing the inner (higher) spikelet cluster of the fruit bunch and shortening the remaining spikelets (Figure 4). This simplifies the thinning operation, yet it can still take up to 3.5 hours to thin a single mature (Bar-Shira, et al., 2019). Furthermore, thinning must be done within a four week period, starting 4-5 weeks after pollination. Yet, many farmers with large orchards start their thinning early since fruitlets thinning is time consuming. An early start may lead to over-thinning or to a requirement to perform another thinning operation to adjust the fruitlet load (Cohen & Glasner, 2015). Automation of the thinning operation is imperative for enabling fruitlets thinning of larger plots with reduced manpower within the desired time frame. Still, due to task complexity, there are currently no commercially available thinning autonomous robot systems (Bar-Shira, et al., 2019). Furthermore, the annual crop expected in 10 years from the currently planted trees is expected to reach more than 50,000 tons. This addition of about 40% of the crop will require an additional 80,000 days of work of thinning, this is without considering the added crop expected from new planting during the period (The plant council, 2019). These forecasts accentuate the necessity of automation.

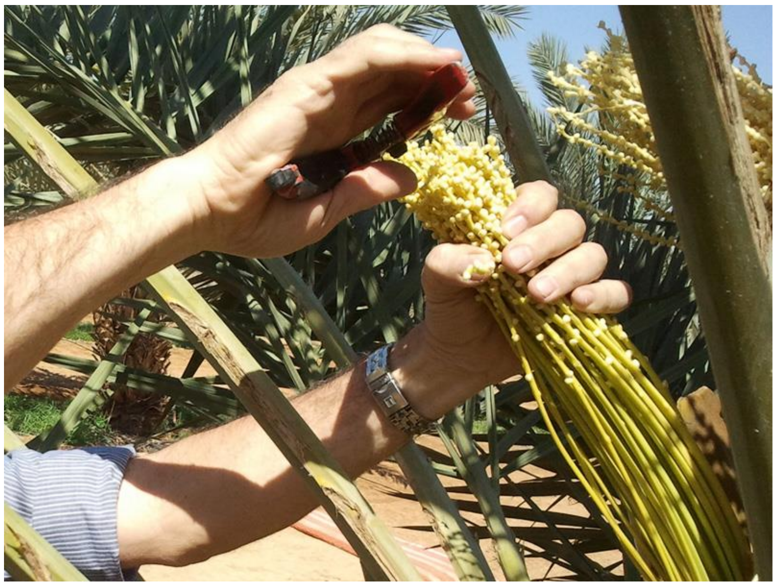


Figure 4: Thinning of a Medjool date fruit bunch. The worker bundles the spikelets with one hand and shortens the spikelets near the bundling point using pruning shears (News from the grove, 2011).

2.3 Policy representation

2.3.1 Movement primitives

Humans perform complex motor tasks in uncertain and changing environments with little apparent effort. They can generalize and therefore adapt to new tasks based on their motor abilities (Mülling, et al., 2013). Human movements seem to be composed of movement primitives. Once learned, they allow humans to quickly adapt their movements to variations of the situation without the need of re-learning the complete movement (Kober, et al., 2012). For example, the overall shape of table tennis forehands is very similar when the swing is adapted to varied trajectories of the incoming ball and different targets on the opponent's court. To accomplish such behavior, the human player has learned by trial and error how the global parameters of a generic forehand need to be adapted due to changes in the situation (Kober, et al., 2012; Mülling, et al., 2013). Understanding how this motor competence may be represented and implemented in robots is of great value.

In robotics, the ability to generate a variety of complex movements cannot be attained by simply storing all movements and recalling them when needed. This approach would be inadequate because the number of possibilities is immense. To achieve its full competence, the motor system must be capable of generalizing beyond the movements that it has experienced in the past. It is possible to do so with movement primitives. Every movement, discrete or rhythmic, can be represented through a sequence (repeated sequence) of a finite number of movement primitives (Mussa-Ivaldi, 1999). To deal with complex movement, one can build a library of movements primitives out of which complex movement can be composed by sequencing. For example, the library may contain a grasping, placing, and releasing movements. Each of these movements is a different movement primitive and is labeled accordingly. For moving an object on a table, a grasping-placing-releasing sequence is required, and the corresponding primitives are recalled from the library (Pastor, et al., 2009).

The movement algorithm developed in this work is based upon dynamic movement primitives (DMPs), a commonly used form of movement primitives (Cohen, Bar-Shira, & Berman, 2019).

2.3.2 Dynamic movement primitives

The problem of learning a mapping between world state and actions lies at the heart of many robotics applications. This mapping, also called a policy, enables a robot to select an action based upon its current world state (Argall, et al., 2009). The goal of motor control and motor learning can generally be formalized in terms of finding a task-specific control policy:

$$u = \pi\,(x, t, \alpha) \qquad (1)$$

that maps the continuous state vector x of a control system and its environment, possibly in a time t dependent way, to a continuous control vector u. The parameter vector α denotes the problem specific adjustable parameters in the policy π. A control policy is supposed to take the motor system from an arbitrary start point to the desired behavior. The desired behavior might be simply reaching a goal point or a cyclic movement, like walking, swimming, chewing, etc. Both cases can be thought of as attractor dynamics, i.e., either a point attractor or a limit cycle attractor.

Systems with attractor dynamics can generally be written as a differential equation:

$$\dot{x} = f(x, t, \alpha) \quad (2)$$

which is almost identical to Equation 1, except that the left-hand-side denotes a change-of-state, not a motor command. Such a kinematic formulation is suitable for motor control if this dynamic system is conceived as a kinematic policy that creates kinematic target values (e.g., positions, velocities, accelerations), which subsequently are converted to motor commands by an appropriate controller (Schaal, et al., 2005; Pastor, et al., 2013). It had become a common practice to model natural phenomena with systems of coupled nonlinear differential equations that exhibit rich abilities for forming coordinated patterns without the need to explicitly plan or supervise the details of such pattern formation (Ijspeert, et al., 2013).

DMP framework was originally proposed in 2001 (Ijspeert, Nakanishi, & Schaal, 2001) and was further extended throughout the years (Schaal, Mohajerian, & Ijspeert, 2007; Hoffmann, et al., 2009; Ijspeert, et al., 2013). DMP is a set of nonlinear equations used for encoding motion policies. DMP equations enable to transform well-understood simple attractor systems with the help of a learnable forcing function term into a desired attractor system. First, a model of a goal-oriented baseline locomotion (i.e. movement primitive) is required. After this baseline model has been accomplished, it can be used to account for more complex phenomena with the help of the coupling dynamics of nonlinear systems and adjusting the DMP parameters (Ijspeert, et al., 2013).

Example for the DMPs equations is listed below. First is the transformation system, instantiated by the second order dynamics:

$$\tau \ddot{y} = \alpha_z (\beta_z (g - y) - \dot{y}) + f(x) \quad (3)$$

where g is a known goal state, α_z and β_z are time constants, τ is a temporal scaling factor that control the duration of the movement, f is a forcing term, and $y, \dot{y}, \ddot{y}$ correspond to the desired position, velocity, and acceleration generated.

The name of the equation is due to its ability to transforms the simple dynamics of the unforced system into a desired nonlinear behavior. Without the function f, Equation 3 is nothing but a linear spring-damper, and, after some reformulation, the time constants α_z and β_z have an interpretation in terms of spring stiffness and damping. For appropriate parameter settings and $f = 0$, these equations form a globally stable linear dynamic system with g as a unique point attractor, which means that for any start position the system would reach g, just like a stretched spring, upon release, will return to its equilibrium point. However, setting f to be a nonlinear function will allow trajectories that are almost arbitrarily complex on the way to the goal g.

Next, there is the canonical system:

$$\tau\dot{x} = -\alpha_x x \qquad (4)$$

where α_x is a constant. From any initial conditions, the canonical system can be guaranteed to converge monotonically to zero. This monotonic convergence of x becomes a substitute for time, x can thus be conceived as a phase variable, where $x = 1$ would indicate the start of time evolution and x close to 0 means that the goal g has been achieved. The name of the equation is because it models the generic behavior of the model equations, a point attractor in this case (for a limit cycle attractor, the canonical system needs to be represented in a different way).

The forcing term f is given in Equation 5:

$$f(x) = \frac{\sum_{i=1}^{N} \psi_i(x) w_i}{\sum_{i=1}^{N} \psi_i(x)} x(g - y_0) \qquad (5)$$

With N exponential basis functions $\psi_i(x)$,

$$\psi_i(x) = \exp\left(-\frac{1}{2\sigma^2}(x - c_i)^2\right) \qquad (6)$$

where σ^2 and c_i are constants that determine, respectively, the width and centers of the basis functions, w_i are the weights of the basis functions that can be adjusted using learning algorithms in order to produce complex trajectories before reaching g, and y_0 is the intial state $y_0 = y(t = 0)$. The phase x appears also multiplicative in Equation 5 such that the influence of f vanishes at the end of the movement when x has converged to zero (Ijspeert, et al., 2013; Pastor, et al., 2013).

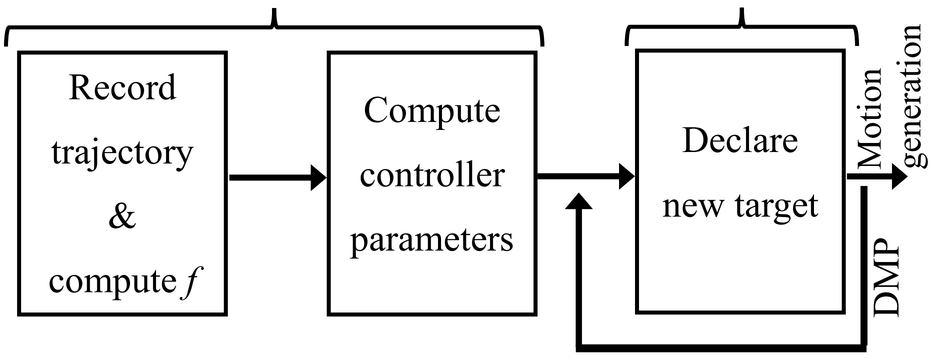
The parameters of a DMP can be divided into three categories: shape parameters, meta parameters, and external parameters. The Shape parameters (w_i) define the spatio-temporal shape of the movement. The meta parameters are the values that adapt the movement behavior, e.g., the start and goal positions (y_0 and g), movement duration (τ) etc. The external parameters determine external, high level, environmental issues that affect DMP operation, e.g., when to start motion with respect to additional task constraints (Kober, Oztop, & Peters, 2011; Kober, et al., 2012).

DMPs have important advantages for planning and controlling a movement. (1) DMP equations do not encode a single, specific desired trajectory but rather a whole attractor landscape. This provides the ability to smoothly recover from perturbations, which is an important property when moving in a dynamic environment. (2) Avoiding the explicit time dependency simplifies the ability of coupling with other dynamical systems. (3) DMP equations allow fast learning of a trajectory with a relatively low number of parameters. The number of basis functions (hence of parameters) can be adjusted depending on the desired accuracy of the fit, with more functions allowing the representation of finer details of movement. Furthermore, these parameters can be obtained efficiently. (4) The ease of modification is also ensured by having the goal state of the movement explicitly encoded into the system. (5) The representation of $y, g, x,$ and τ in the DMP equations result in an invariant policy under transformations of the initial position, the goal, the amplitude, and the duration (Ijspeert, Nakanishi, & Schaal, 2002; Kober, Oztop, & Peters, 2011).

In the DMP formulation the shape parameters are calculated a-priori and retained for motion generation during run time (Figure 5). This limits the ability of each DMP to represent the characteristics of the original trajectory to relatively simple trajectories, i.e., trajectories with limited changes in curvature and jerk, since big changes, e.g. starching the learned trajectory, decrease the accuracy of the generated movement. As a result, the movement primitives coded by DMPs are relatively simple, thus a large set of DMPs is required for representing complex trajectories (Cohen & Berman, 2013). The method called Tight dynamic movement primitives (TDMP) addresses this weakness.

2.3.2.1 Tight dynamic movement primitives

TDMP is an adaptation to the DMPs. In the TDMP formulation, the stored movement parameters are the coordinates of the typical motion trajectory. The coordinates of the path are linearly transformed during run time based on the required meta parameters, and the suitable shape parameters are calculated from the transformed trajectory using linear regression analysis (Figure 5). TDMP requires less primitives, with respect to DMP, to compose a complex trajectory and it is independent from specific coordinates. Furthermore, it allows a more accurate fit to the shape of the trajectory (Cohen & Berman, 2013).


Record trajectory & compute f
Compute controller parameters
Declare new target
Motion generation
DMP

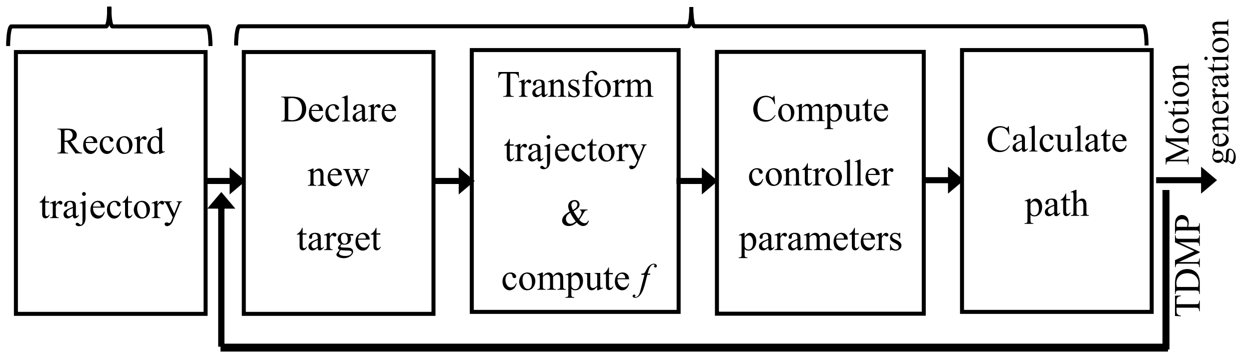


Figure 5: Dynamic movement primitive flowchart (learning and motion generation process). Top: DMP, Bottom: TDMP (adapted from Cohen & Berman, 2013).

2.3.3 Generalizing DMP to new tasks with meta parameters adaptation

The movement executed with DMP can be adapted both spatially and temporally without changing the overall shape of the motion. However, with current techniques, in many cases, the robot needs to learn a new elementary movement even if a parametrized motor plan exists that covers a related situation. The cost of experience is high as sample generation is time consuming and often requires human interaction (e.g., in cartpole, for placing the pole back on the robot's hand) or supervision (e.g., for safety during the execution of the trial). Given that the behavior will not drastically change between related situations, it would be better to generalize the learned behavior to the modified task. Generalizing a teacher's demonstration or a previously learned policy to new situations may reduce both the complexity of the task and the number of required samples (Kober, Oztop, & Peters, 2011; Kober, et al., 2012).

Several methods have been suggested for handling a change in task requirements during run time. These include: augmentation of the traditional DMP approach with a memory of sensory events that occurred during the learning of the movement primitive and using the additional information to create a task policy with corrective commends in run time (Pastor, et al., 2013), and creating a task policy that is composed of several movement primitives weighted by their ability to generate successful movements in the given task context (Mülling, et al., 2013). Such adjustments are bounded to a small region about the original parameter values.

To increase the ability to generalize the movement primitive to a larger continuum of tasks, one can exploit the influence that the DMPs parameters have on the movement. Previous studies showed that adapting the meta parameters can help achieve generalization of the movement behavior (Kober, Oztop, & Peters, 2011; Kober, Wilhelm, Oztop, & Peters, 2012).

Previous work was done on adapting meta parameters with reinforcement learning (RL). Some researchers developed methods of learning meta parameters for given shape parameters (Kober, Oztop, & Peters, 2011). Others have integrated learning meta and shape parameters (Tamosiunaite, et al., 2011; Stulp & Schaal, 2011). While RL indeed facilitates acquiring new skills, it has several limitations. The continuous parameter space along with the high dimensionality of typical robotic manipulator motor-learning problems, might lead to dimensionality reduction that severely limits the dynamic capabilities of the robot. This may be unacceptable for many tasks (Kober, Bagnell, & Peters, 2013).

A different body of research, of constructing data-driven models, seeks to generalize available data, forming concise representations that capture salient motion characteristics. Researchers suggest using hierarchical Bayesian models (HBMs) for estimating both meta and shape parameters for probabilistic movement primitives (Rueckert, et al., 2015). Probabilistic movement primitives facilitate encoding optimal behaviors in stochastic systems, yet they are less appropriate when deterministic system behavior is required.

The current work is a part of an effort for forming a nonlinear and deterministic mapping from task requirements to DMP meta parameters. The mapping is based on a representation of the task and meta parameter manifold (Lee, 2010) with a deep neural network.

2.4 Deep neural networks

2.4.1 Overview

Deep neural networks are computational models that can acquire and maintain knowledge. They can be defined as a set of processing units, represented by artificial neurons, interlinked by a lot of interconnections, implemented by vectors and matrices of weights. One of the most relevant features of neural networks is their capability of learning from the presentation of samples and generalize solutions, to wit, the network can produce an output which is close to the expected output of any given input values. There are different architectures of DNNs. One of the most versatile architectures is the multilayer perceptron (MLP) network that belong to the multiple layer feedforward architecture (the information always flows in a single direction), whose training is performed with a supervised process. In supervised learning each training sample is composed of the input attributes and their corresponding outputs. It is from this information that the network will formulate hypothesis about the system being learned. MLP networks are commonly utilized for function approximation (Da Silva, et al., 2017), as required for forming a mapping between task requirements to DMP meta parameters.

2.4.2 Data acquisition for training deep neural networks

DNNs are known for their abilities to solve complex real-world problems (Chiroma, et al., 2018). Yet, they are notoriously known for requiring large training datasets for attaining an accurate representation. Nowadays, with the support from developing computational power, big data analysis using neural networks has achieved great success. large volume of big data provides tremendous training samples which enable the training of neural networks with large number of parameters (Zhang, Guo, & Wang, 2017).

The cost of generating labeled data for training DNN is often an obstacle. Acquiring big amount of data is especially challenging when entering a new field where there are no previous datasets (Ganin, et al., 2016). Establishing dataset for the current problem was an immense challenge as the thinning epoch is short and in this short time the structure of the fruit bunch changes significantly. Furthermore, the data acquisition is complex since an altitude tool is needed in order to reach the fruit bunches within the date crown, along with security arrangements, professional accompaniment, and time. To overcome this challenge, the power of the simulation was exploited. Computer simulations are computer-generated, dynamic models of the real world and its processes (Smetana & Bell, 2012) that enables creation of myriad realistic artificial data that facilitate increased performance of the DNN that comes with large datasets (Barth, et al., 2018), thus, improved ability to–generalize to unseen data is attained (Varga & Bunke, 2003; Jaderberg, et al., 2014).

2.4.3 Adapting motion parameters with deep neural networks

For a given robotic system, environment, and task, there are n task parameters that represent the task requirements, $\delta_1, \dots, \delta_n$, m meta parameters, $\gamma_1, \dots, \gamma_m$, and p shape parameters, $w, \dots, w_p$. The task and meta parameters are related through a manifold $\emptyset(\delta_1, \dots, \delta_n, \gamma_1, \dots, \gamma_m)$ embedded in the parameter space. Neural networks are suited for efficiently representing manifold structures (Basri & Jacobs, 2017) and thus for forming a mapping between the task and meta parameters. The hierarchical, layered structure of deep neural networks facilitates capturing complex data structures where the outputs of such networks are continuous piecewise linear functions of the inputs (Basri & Jacobs, 2017). The neural network can take as input the vector of task parameters and supply as output the vector of suitable meta parameters.

Neural networks have hyperparameters whose values can be tuned to improve performance. These include: the number of hidden layers, the number of neurons in each layer, the batch size (the number of input samples processed before updating model parameters), the learning epochs (the number of times that the learning algorithm will undergo the entire training dataset), the loss function (the function used to calculate the gap between the correct scores and the scores computed by the neural network based on its current weights), and the optimizer (the gradient descent optimization algorithm). Furthermore, different optimizers have different parameters whose values can be tuned as well (Ruder, 2016; Sze, Chen, Yang, & Emer, 2017). Hyperparameters have great impact on the performance of the model, effecting both learning run time and final accuracy attained. Accordingly, a variety of hyperparameter optimization methods were developed (Bergstra & Bengio, 2012; Hoos, Ca, & Leyton-Brown, 2014; Xu, 2015). Based on an analysis of the parameter manifold we performed a grid-search (Bergstra & Bengio, 2012) for central hyperparameters determining network structure.

Chapter 3: Constructing a synthetic annotated dataset of Medjool date fruit bunches

3.1 Overview

To cope with the absence of annotated data of Medjool date images, a stochastic model that captures the geometry of a fruit bunch was devised and a 3-dimensional visualization was composed based on the model. Medjool date experts were part of the model derivation and took part in its structure validation. The model enabled generation of datasets required for training and testing the neural network.

3.2 The model

3.2.1 Forming the model

The structure of the fruit bunch was captured with an assembly of geometrical shapes. The rachis has elongated elliptical shape that narrows towards it end. The spikelets have elongated shape and can be modeled by constructing a geometric spatial curve. The fruitlets have an oval shape and are relatively small with respect to the spikelet length, therefore can be modeled with a stereotypic oval shape. Accordingly, the fruit bunch was modeled with the following geometrical shapes: the rachis is represented as an elliptical cylinder that a cone is placed on top of it, the spikelets are Bezier curves (Singh, 1995), and the fruitlets are spheres.

During the formation of the model, we consulted with two physiologists, experts in Medjool date cultivation. They expressed their professional opinions, advised, and referenced to literature when needed. The professional accompaniment was mostly done in the form of face-to-face meetings, a total of 8 meetings were held. Correspondence by email existed when necessary. The model development considered the environment in which the system will operate (in order to enable the system to achieve good performance in a physical environment, the training of the DNN must be on a realistic data) and the end-user (much emphasis was placed on creating a visualization that the end-user, i.e. the farmer, can relate too. by building a model that the end-user can relate to, we are increasing his willingness to use to final product). The forming of the model began with investigating a sample of Medjool date fruit bunches and the fruit bunch parts refined with the help of the physiologists. A preliminary sketch containing the main components was drawn and a visualization was performed in python OpenGL (Shreiner, et al., 2013) (Figure 6). During the establishment of the model we discovered a great variety between fruit bunches from different whorls and from different points of time throughout the thinning period. As a result, the distributions of the model's parameters were fitted depending on the whorl and the thinning period (the thinning period was divided to 3 epochs: early, medium, and late). A total of 9 combinations were developed in parallel.

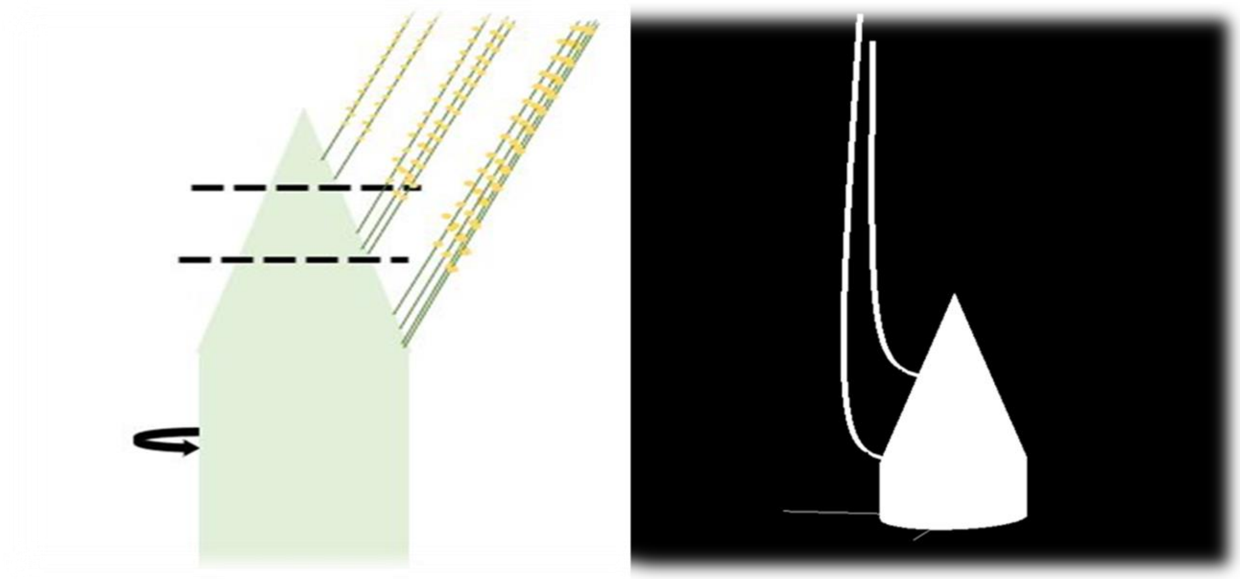


Figure 6: A sketch of a Medjool date fruit bunch (Left: word, Right: python OepnGL).

3.2.2 The stochastic model

Eighteen parameters were defined in the model to capture the geometry of the fruit bunch. The parameters represent main features of the fruit bunch without dependency on the whorl and thinning epoch. The parameters define the rachis (width, depth, length, bending, and orientation), the spikelets (quantity, dispersion, and size), and fruitlets (quantity, dispersion, size, and the natural fallout probability). A probability function was defined for each of the parameters and distribution parameters were fitted for the three thinning epochs and the three whorls (9 sets of probability parameters). The parameters, the distributions, and the distribution parameter values were established based on interviews with plant experts and farmers, visual comparison of the synthetic images with images of physical fruit bunches, and graphical considerations.

Table 1 present the Medjool date fruit bunch model parameter distributions. Distribution parameters given in parentheses for the 1st epoch-1st whorl. The full model is detailed in Appendix I.

Table 1: Medjool date fruit bunch model parameter distributions. Distribution parameters given in parentheses for the 1st epoch-1st whorl.

Index	Parameter	Probability function[a]
p_1	Rachis width [cm]	$U(a,b),(2{,}3)$
p_2	Rachis depth [cm]	$U(a,b)*(p_1),(0.56{,}0.76)\times(p_1)$
p_3	Viewed section of rachis [cm]	$Constant, 20$
p_4	Hidden rachis length [cm]	$Triang(a,b,c), 20, 25, 30)$
p_5	Angle between rachis to tree trunk [°]	$U(a,b),(0, 10)$
p_6	Number of spikelets	$Triang(a,b,c),(80{,}85{,}100)$
p_7	Number of spikelets clusters	$Constant, 3$
p_8	Distance of spikelets clusters from top of presented rachis	$Bottom\ cluster$: $0\times(p_4)$ $Middle\ cluster$: $0.4\times(p_4)$ $Top\ cluster$: $0.6\times(p_4)$
p_9	Dispersion of spikelet quantity between the clusters	$Bottom\ cluster$: $0.5\times(p_6)$ $Middle\ cluster$: $0.3\times(p_6)$ $Top\ cluster$: $0.2\times(p_6)$
p_{10}	Spikelet length [cm]	$Triang(a,b,c),(75, 80, 100)$
p_{11}	Spikelet radius [cm]	$U(a,b),(2{,}2.5)$
p_{12}	Clamps width (i.e. the largest distance between two spikelets) [cm]	$Triang(a,b,c),(25, 27, 35)$
p_{13}[b]	Number of fruitlets per spikelet (on bottom cluster)	$U(a,b),(60{,}70)$
p_{14}	Distance from the spikelet base to the first fruitlet [cm]	$Triang(a,b,c),(20{,}27{,}45)$
p_{15}	Fruitlet radius [cm]	$Triang(a,b,c),(1, 1.2, 1.5)$
p_{16}[b]	Distance between two adjacent fruitlets [cm]	$Exp(\frac{1}{\lambda}),(1.3)$
p_{17}	Natural fallout probability [%]	$Berr(p), 16$
p_{18}	Bending of the fruit bunch [%]	$Constant, 0$

$U(a,b)$ – uniform distribution; $Triang(a,b,c)$ – triangular distribution; $Berr(p)$ – Bernoulli distribution; $Exp(\frac{1}{\lambda})$ – exponential distribution. (b) These parameters were used for testing the outcome of the generation process detailed blow.

3.2.3 Visualization of the model

The visualization was developed with python 3.7. The libraries used and the main methods are detailed in Appendix II. The spikelet clusters are modeled as ellipses along the cone, parallel to its base. They are located at three heights according to parameter p_8. The spikelets are drawn in 3° intervals around each cluster, where each spikelet has an appearance probability that is calculated according to parameters p_6 and p_9. Parameters p_{10}, p_{12} and p_{18} are used to define the four Bezier points defining the cubic Bezier curve for each spikelet,

$$B(t) = [t^3\ t^2\ t\ 1]\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} \mathrm{V}_0 \\ \mathrm{V}_1 \\ \mathrm{V}_2 \\ \mathrm{V}_3 \end{bmatrix},\ 0 \le t \le 1 \tag{7}$$

Where V_0, V_1, V_2, and V_3 are Bezier points in the 3-dimensional space $V_i = [V_{i,x},\ V_{i,y}, V_{i,z}]$. The curve starts at V_0, that lies on the circumference of the cluster, and ends at V_3. V_1 and V_2 are the Bezier control points. For all spikelets, $V_{3,y}$, i.e., the height of V_3, is defined by,

$$\mathrm{V}_{3,\mathrm{y}} = (p_3 + p_{10}) \times p_{18} \tag{8}$$

This causes the spikelets in the higher (inner) clusters, i.e. the middle and top clusters, to be shorter than the spikelets in the lower (outer) cluster and achieves the typical overall shape of the cluster which bears resemblance to that of a broom.

For spikelets in the lower cluster of fruit bunches from the 1st epoch, the x and z coordinates of V_3 are depended on the parameter p_{12} and on $V_{0,j}$ (j=x, z) such that the point V_3 lies at the same angle as V_0 and in a radius that equals $0.5 \times\ p_{12}$ (for spikelets found in the higher clusters, the point V_3 is adjusted in a similar way with a smaller radius). For $Berr(0.7)$ of the spikelets, the two control points, V_1 and V_2, are defined by constant values with addition of white noise that was set empirically. Hence, their curvature is typical for the whorl-epoch combination and is constrained within the overall structure defined by the model. For the remaining spikelets, $V_{1,y}$ and $V_{2,y}$ has a considerably larger noise value to capture natural disorder found in the field.

For spikelets in all clusters of fruit bunches from the 2nd epoch and the 3rd epoch, V_1 is depended on V_0 such that the point V_1 lies at the same angle as V_0, and at a higher height which was empirically set. The x and z coordinates of V_3 and V_2 lie at a 90° interval. $V_{2,y}$, i.e., the height of V_2, is defined by,

$$\mathrm{V}_{2,\mathrm{y}} = (p_3 + p_{10}) \tag{9}$$

For $Berr(0.7)$ of the spikelets, $V_{1,j}$, $V_{2,j}$,and $V_{3,j}$ (j=x, z) are defined by constant values with addition of white noise that was set empirically. Hence, their curvature is typical for the whorl-epoch combination and is constrained within the overall structure defined by the model. For the remaining spikelets, those coordinates have a considerably larger noise value to capture natural disorder found in the field.

The fruitlets are drawn in spiral about the Bezier curve. The first fruitlet is drawn at the first point whose distance from the starting point is larger than parameter p_{14}. From that point and until the end of the curve, fruitlets are drawn at each curve point with probability according to parameter p_{17}. For the 1st whorl-1st epoch each Bezier curve consists of 250 points, such that the number of fruitlets per spikelet (p_{13}) and the distance between fruitlets (p_{16}) fit the model.

Figure 7 presents examples for two pairs of physical and synthetic fruit bunches from different whorl-epoch combinations.

3.2.4 The length of the remaining spikelets

Nowadays, the length of the spikelets after thinning is determined based on the time of thinning, the fruit bunch's whorl, and by the farmer long term yield policy which may be influenced by various parameters including the fruitlet load, the expected natural fruitlet fallout, or the girth of the rachis. In order to mimic such a decision-making process a DNN is trained for each combination of whorl and epoch (a total of 9 networks). For training of the DNN, the ground truth needs to be known as it is a supervised learning algorithm (Da Silva, et al., 2017). Since the thinning method implemented in the work is shortening of the fruit bunch, the DNNs output is the remaining length of the spikelets after the thinning. The spikelets remaining length was quantified with respect to the thinning epoch, the whorl, and thinning technique implemented in the current work (e.g., for fruit bunches from the 1st epoch-1st whorl, based on a goal of 400-500 fruits at harvest, the remaining spikelet length for each fruit bunch was set as the average distance from the base of the spikelets in the lower (outer) cluster to their 12th fruitlet).

3.3 Validation of the model

The final tuning and validation of the model were made for fruit bunches from the 1st whorl-1st epoch. A dataset of ten fruit bunches was generated. The database contained two images of each object, with one of the images showing the cutting location of the fruit bunch. Additionally, properties of each fruit bunch (the fruitlets load and the number of the remaining fruitlets after thinning) were detailed in an accompanying file (Figure 8). Both physiologists were asked to review and validate the dataset visually and numerically. The dataset was approved and datasets for training and testing the neural network were generated.

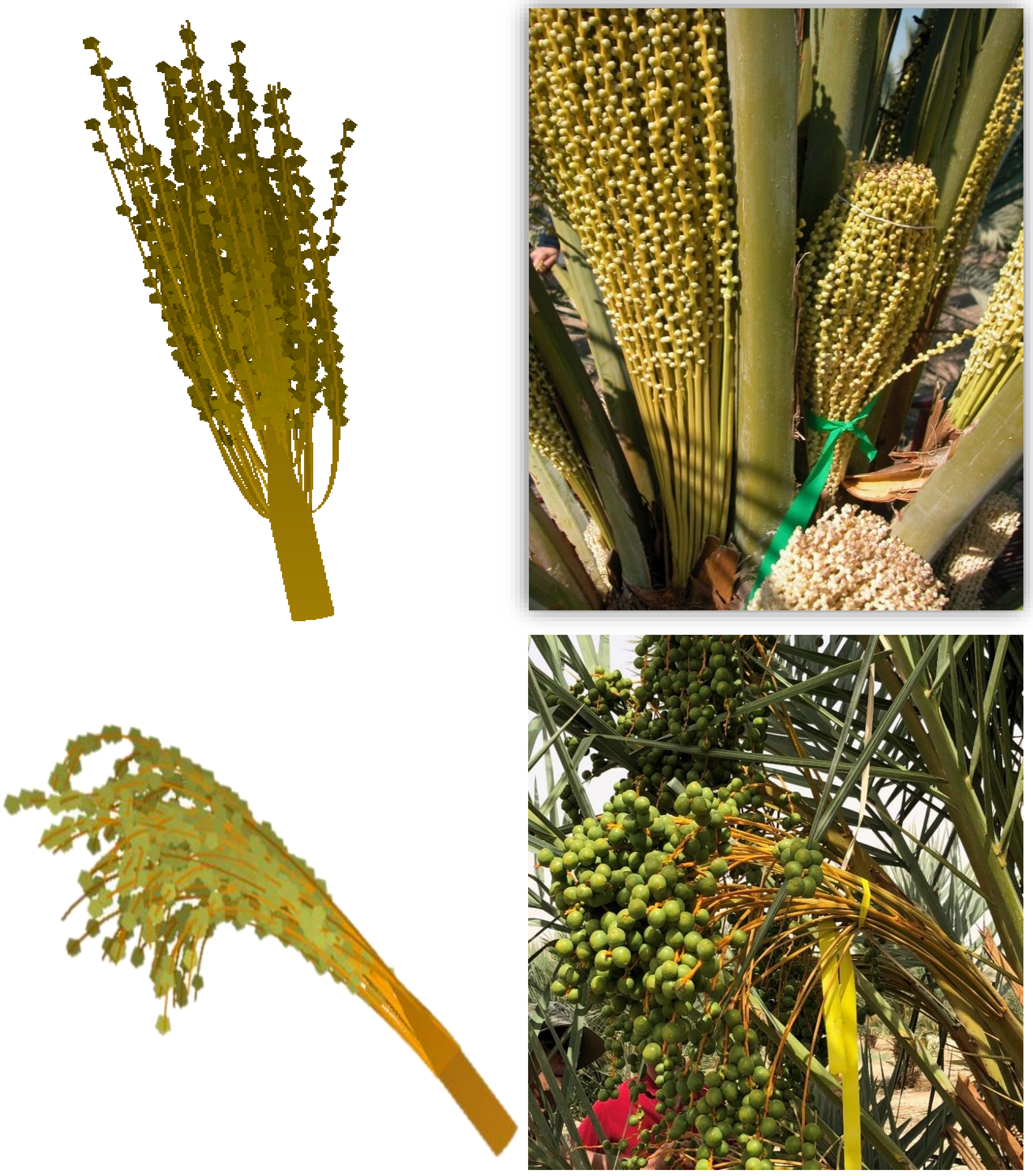


Figure 7: Medjool date fruit bunches (Left: synthetic, Right: physical image). Top: 1st epoch-1st whorl; Bottom: 3rd epoch-2nd whorl *(Bar-Shira, et al., 2019)*.

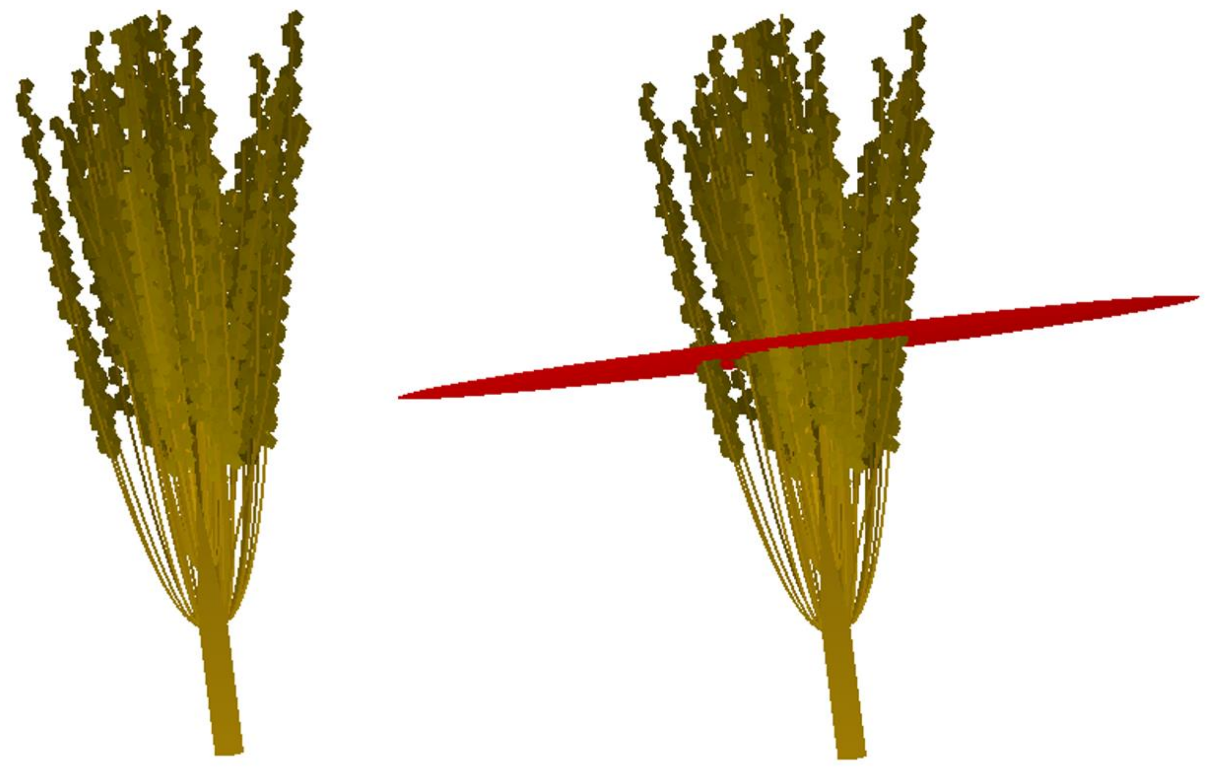


Figure 8: Marked Medjool date fruit bunch from the 1st epoch-1st whorl. Fruitlets load: 3290; Fruitlets after thinning: 804.

Chapter 4: Learning motion parameters

4.1 Complexity analysis of task and meta parameter manifold structure

Examining the complexity of the interactions between the task and meta parameters is important for understanding the manifold structure through which these parameters are connected. We performed complexity analysis using two methods, principal component analysis (PCA) and locally linear embedding (LLE). PCA is a classical method often used for dimensionality reduction. Based on the dataset it determines a coordinate system which is a linear transformation of the original data features comprising the data's principal components (PCs), i.e., the axes onto which the retained variance under projection is maximal. PCA enables representation of dominant patterns in the data by plotting the primary principal component (this relies on the assumption that the directions of maximum variance express most of the information in the data) (Wold, Esbensen, & Geladi, 1987). LLE (Roweis & Saul, 2000) computes low-dimensional, locally linear, neighborhood-preserving embedding of high-dimensional inputs. The method is based on simple geometric intuitions: if data is sampled from a smooth manifold, then neighbors of each point remain nearby and are similarly co-located in the low-dimensional space. In LLE, each point in the dataset is linearly embedded into a locally linear patch of the manifold. Then low-dimensional data is constructed such that the locally linear relations of the original data are preserved (Wang, 2012). The downside of using LLE is that, unlike for PCA analysis, the relationship between the output and input parameters is not readily available for analysis. Both concepts facilitate visualization of the parameter response surface in low dimensional space (PCA for data which contain a linear or near linear structure and LLE for non-linear data structures) (Liu, Weissenfeld, & Ostermann, 2006).

4.2 Learning task and meta parameter mapping

The suggested workflow has additional steps during learning in order to learn the task and meta parameter manifold and again during runtime in order to obtain the meta parameters with the learned mapping. The workflow consists of the following steps (Figure 9): data containing tuples of related task and meta parameters is established in simulation, a complexity analysis is performed to acquire intuition regarding the task and meta parameter manifold structure and the manifold is learned. Then, as the TDMP require, the movement primitive trajectory is recorded prior to receiving a new task where the features of the trajectory are points in the 3-dimensional space along the fitted curve that reflect spatial information, i.e., the shape of the movement. When receiving a new task, the task parameters are determined, they are given as an input to the learned mapping, and the suited meta parameters are obtained. Next, the movement primitive trajectory is adapted according to the new parameters, the control (i.e. shape) parameters are computed analytically, and the movement is generated.

The task and meta parameter mapping for adapting the DMP parameters during run-time, is learned a-priori (Figure 9). It possible to learn the manifold with different methods. Two methods, deep neural networks and kernel regression were tested in our group (Cohen, Bar-Shira, & Berman, 2019). In the current work we elaborate on using a deep neural network for learning this mapping. The neural network was constructed considering the complexity of the structure of the task and meta parameter manifold.

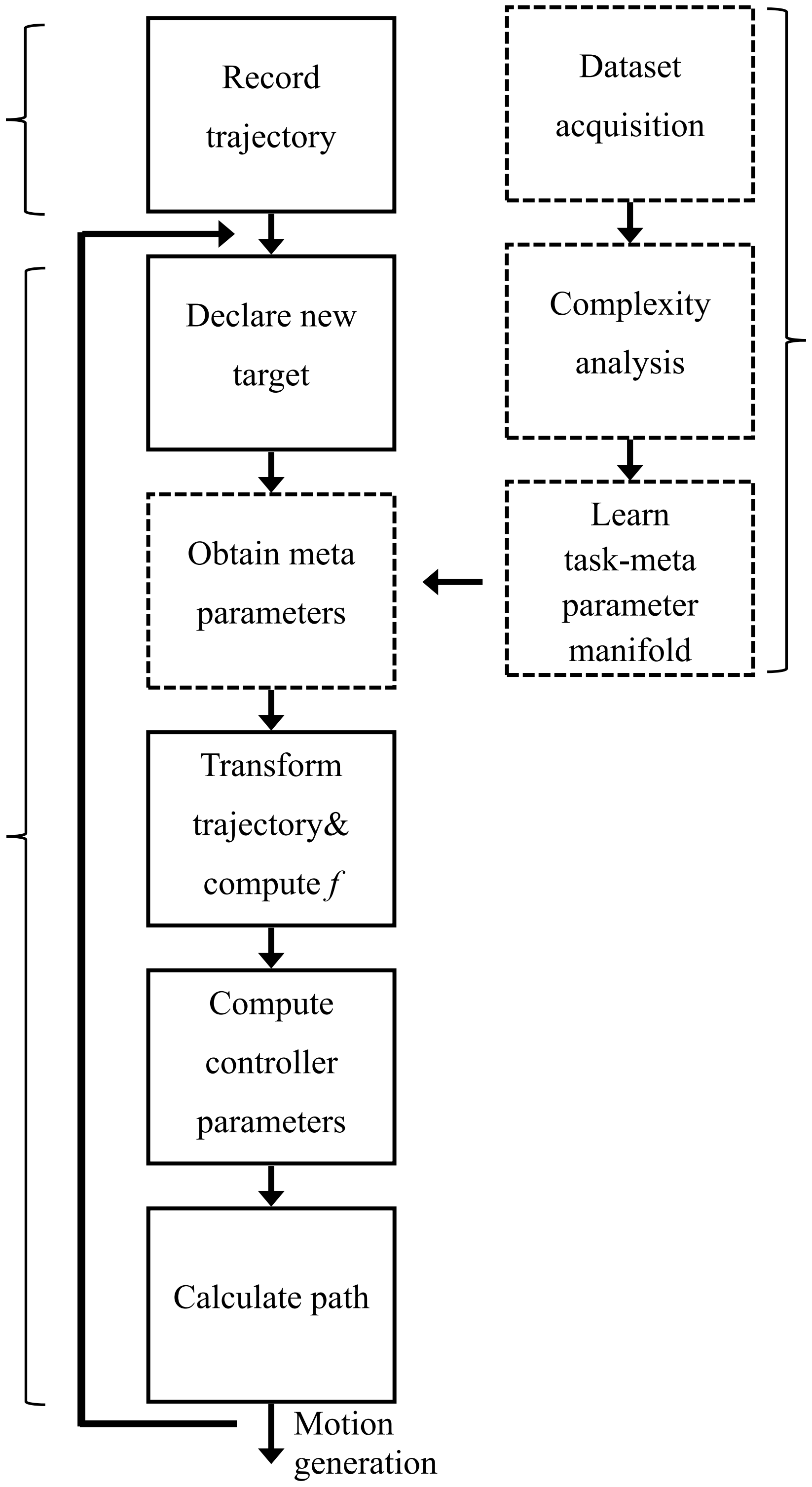


Figure 9: Dynamic movement primitive adapted flowchart. New steps are surrounded by a dashed box.

Chapter 5: Experiments

5.1 Overview

Two experiments with two different tasks were held for examining the adapted DMP methodology: robotic soft ball throwing and Medjool thinning. In the robotic soft ball throwing experiment, the mapping between task requirements to meta parameters was learned on a synthesized data. In the Medjool thinning experiment the mapping was learned based on the Medjool date fruit bunch model. The ball throwing experiment is a simpler task and it was conducted as a first toy example for initial tests of the methodology. The Medjool date thinning task is more complicated and is an example of a practical task for which the methodology is suited.

5.1.1 Robotic soft ball throwing

A ballistic movement of a robotic soft ball throwing task was chosen since while the motion trajectory can be readily defined for the throwing motion, the task dynamics for a soft ball contain non-linear components. Thus, setting the exact parameters for the trajectory is not non-trivial. For robot soft ball throwing, the task requirement is to throw a ball to a specific goal positioned on a plain (i.e. the floor). The task parameters, describing the required landing point on the floor, are the radius r and the horizontal angle α (Figure 10). The meta parameters, describing the endpoint of the robotic throwing motion, are the horizontal angle β, height h, radius R, and the motion duration τ (Figure 10). The velocity profile of the robot is trapezoid (maximal acceleration, zeros acceleration, and maximal deceleration). To achieve the required task the ball is released not at the end of the motion, but rather when the robot is moving at its maximal velocity, directly before it starts decelerating.

Two models were considered for the ball throwing task, a ballistic model and an adapted model. The ballistic model was based on standard ballistic motion equations. The adapted model was developed in order to make the simulation more realistic and account for some of the limitations of the physical robot. The adapted model reflects the fact that it is harder for the robot to develop high velocities throwing the ball sideways (larger absolute β values) using the current motion profile and that it is also harder to develop high velocities for small throwing radiuses (smaller R values). The full workflow was examined for both models.

In the robotic soft ball throwing, the ability of the DNN to learn the task and meta parameter manifold and thus achieve an accurate movement was compared with that of kernel regression. Kernel regression is also suitable for learning efficient manifold representations, yet it has different strengths compering to the DNN (Cohen, Bar-Shira, & Berman, 2019). The two learning methods are examined in light of different sizes of training datasets.

5.1.2 Medjool thinning

The Medjool date thinning experiment conducted to test the competence of the DNN to learn the mapping between task requirements to meta parameters for 1st epoch-1st whorl fruit bunches based on the Medjool date fruit bunch model. The task parameters were defined based on farmer account regrading critical parameters for the decision on the spikelets cutting length and based on measurable parameters. These include the fruitlets load (FL) and the parameters of the minimal bounding box surrounding the spikelets (length (LS), width (WS), and angle (AS)) (Figure 10). The meta parameter is the spikelets remaining length, L, i.e., the path length for advancing in direction AS in the frontal plane (with respect to the camera viewpoint) along the spikelets, from the intersection point between the rachis and the spikelets (Figure 10). For the Medjool thinning experiment, the following steps were performed: Datasets acquisition, complexity analysis, learning task and meta parameter manifold, and obtaining the meta parameter for new tasks.

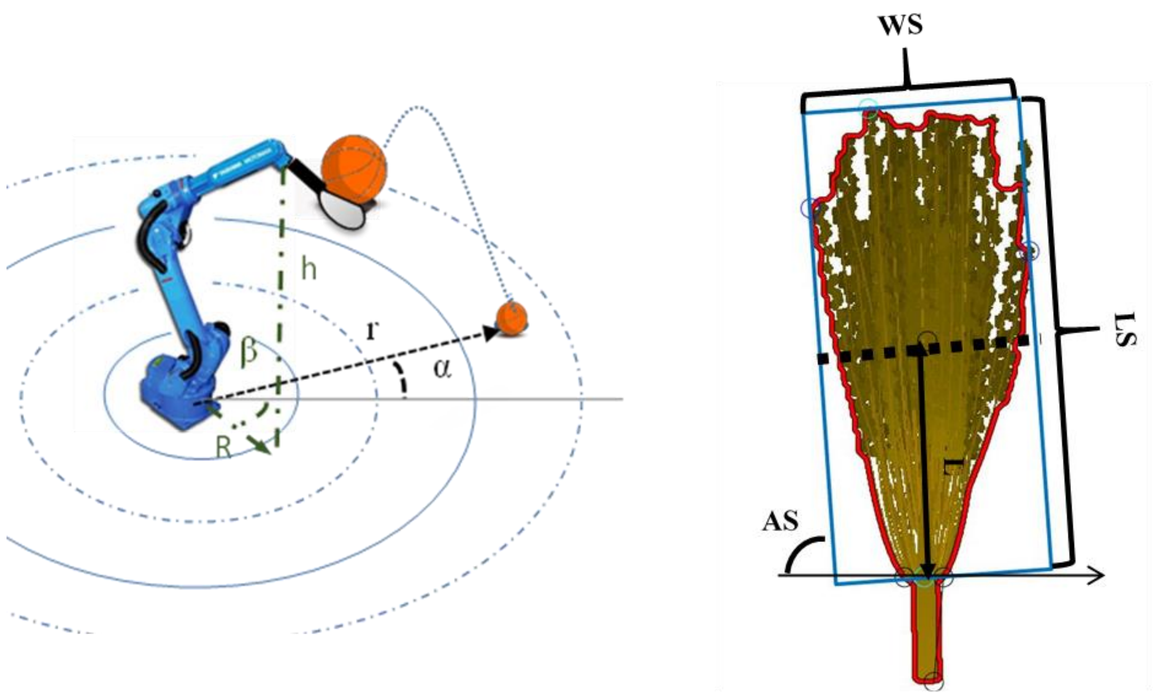


Figure 10: Task and meta parameters illustration. Left: robotic soft ball throwing experiment. The task parameters are the landing point on the floor defined by the radius r and the horizontal angle α. The meta parameters are motion endpoint defined by the horizontal angle β, height h, and radius R (and motion duration τ) (Cohen, Bar-Shira, & Berman, 2019); Right: Medjool thinning experiment. The task parameters are the fruit bunch defined by the length LS, width WS, and angle AS of the minimal bounding box surrounding the spikelets (and the fruit load FL). The meta parameter is the spikelets remaining length L (adapted from Bar-Shira, et al., 2019).

5.2 Data acquisition

5.2.1 Robotic soft ball throwing

Datasets containing tuples of related task and meta parameters were created using two simulation models in Matlab 2015a to account for a 6 degrees of freedom HP6 manipulator (Yaskawa, Motoman, Japan). The simulations differed based on the dynamic model used for computing the ball's trajectory (ballistic model and adapted model). For both models the meta parameter values were randomly sampled from a uniform distribution within the following intervals: β=[-45°,45°], h=[40mm, 140mm], R=[500mm,700mm]. Given the meta parameters and typical trajectory (A typical trajectory suitable for throwing the ball was programmed and tested in the physical environment and then transformed to the coordinates of the simulated environment), shape parameters were calculated using TDMP. Then the resulting task parameters were calculated by using the DMP controller in simulation (ballistic or adapted). The landing point was determined based on the dynamic equations of the ball's trajectory after it was released by the robot.

Different datasets were constructed for different analyses. For all datasets, the robot started at an initial configuration in which the end effector was low and close to the robot's base. Motion duration was set to a constant value of τ=5 since preliminary analysis with the physical robot showed that this is an appropriate value for a duration of a successful shot.

Datasets of different sizes were constructed. 8, 27, 64, 125, 216, 343, 512, 729, and 1000 parameters tuples training sets and 125 parameters tuples test set for each simulation model (ballistic model and adapted model).

5.2.2 Medjool thinning

A training set of 800 tuples and a test set of 200 tuples of related task and meta parameters were constructed. For generating the datasets, fruit bunches from the 1st epoch-1st whorl were sampled from the model. Each fruit bunch was sampled with different seed to create datasets with independent samples. Each sample of a fruit bunch was automatically visualized, and a screen shoot was saved along with the fruit bunch parameters (e.g. the fruitlets load and the spikelets remaining length). The length, width, and angle of the minimum bounding box surrounding the spikelets were computed based on image analysis. The final datasets hold the following parameters: $[FS \pm 10\%, LS, WS, AS, L]$ where the 10 percent error in the fruitlets load is to account for the sonar error.

5.3 Examining data complexity

The relationship between the parameters was examined using both PCA and LLE methods, and a mapping was devised using a neural network. First the PCA was computed and the weights of each parameter in the top principal components were examined. The required dimensionality reduction was set, based on the number of principal components required for explaining 95% of the variance, and the LLE was computed.

The PCA weights influenced the effort to minimize the error of each meta parameter when learning the structure of the neural network. In case of multiple meta parameters, a meta parameter that had larger weights in the top principle components was marked as more important hence the error threshold guiding the training was lower. Problem complexity was assessed by examining the variance explained by the PCA components and the layout of the top components of both PCA and LLE in 3-dimensional figures. When the accumulation of 95% of the variance required more components, the problem space is more complex. When the PCA and LLE parameter dispersion is less uniform problem complexity is higher. Higher problem complexity requires either deeper networks, more neurons in each hidden layer, or both. The code was developed with python in google colaboratory. The libraries used and the flow of the code are detailed in Appendix II.

5.4 Defining network structure and devising a mapping

In both experiments, the number of neurons in the input layer was equal to the number of task parameters and the number of neurons in the output layer was equal to the number of meta parameters. MLP networks were used. The networks consist with a ReLU activation function, mean absolute error loss function, and Adam optimizer (Glorot, Bordes, & Bengio, 2011; Ruder, 2016). The batch size and the number of epochs were empirically set to 32 and 1000 respectively. A grid-search was performed to adjust the number of hidden layers, the number of neurons in each layer, and the learning rate for the optimizer. Based on the analysis of the parameter space we established the boundaries of the grid-search (Bergstra & Bengio, 2012) for adjusting the number of hidden layers and the number of neurons in each layer. Early stopping technique was applied, according to which, the training stops when the performance of the model on the validation set did not improve following subsequent number of epochs (Prechelt, 1998). The final model for a given dataset was devised with the combination of hyperparameters values that minimized the mean absolute error (MAE) between the estimated and truth values, on the validation set,

$$\mathrm{MAE}_y=\frac{1}{n}\sum_{t=1}^{n}|y_t-\hat{y}_t| \qquad (9)$$

Where y is a meta parameter, n is the number of samples, y_t is the truth value, $\hat{y}_t$ is the estimated value.

The code was developed with python in google colaboratory. The libraries used and the flow of the code are detailed in Appendix II.

5.5 Measures

We examined the estimation error of the meta parameters in both experiments. The estimation error of the meta parameters is directly computed from the output of the DNN (i.e., the learned mapping) and the corresponding ground truth values. The error in meta parameter estimation and its relationship to the task parameter estimation error is important for evaluation of the estimation method used for mapping. For the robotic soft ball throwing experiment, we additionally tested the movement error (i.e., the distance between the landing point of the ball to the task requirement), which characterizes the overall performance of the method. For estimating the movement error an additional step was executed to compute the landing point of the ball based on the estimated meta parameters.

For distance estimation (robotic soft ball throwing: h (robot endpoint height), R (robot endpoint radius), r (ball landing point radius). Medjool thinning: L (length of the remaining spikelets)), the error was calculated as the mean scaled absolute error (MSAE) between the estimated and truth values,

$$\text{MSAE}_y = \frac{1}{n}\sum_{t=1}^{n}\frac{|y_t - \hat{y}_t|}{y_t} \qquad (10)$$

Where *y* is one of the parameters (h, R, r, L), *n* is the number of samples, y_t is the truth value, $\hat{y}_t$ is the estimated value.

For the angle measures (robotic soft ball throwing: β (endpoint horizontal angle) and α (ball landing point horizontal angle)), the error was calculated as the mean absolute error (MAE) between the estimated and truth values (Equation 9), Where *y* is one of the parameters (β, α), *n* is the number of samples, y_t is the truth value, $\hat{y}_t$ is the estimated value.

Chapter 6: Results

6.1 Robotic soft ball throwing

In both the ballistic and the adapted model, the top 3 of 5 PCA components explain more than 95% of the variance (Figure 11). Accordingly, the complexity of both models is moderate. The dispersion of the first 3 PCA components was similar and nearly uniform in both models (Figure 13) but the dispersion of the first 3 LLE components differed, showing different concentration along different axes, suggesting stronger internal constraints among the parameters of the adapted model (Figure 13).

Examining the PCA components shows a clear separation between the parameters (Figure 12). The angles (α, β) have very high weights in the 1st component, the distances (r, R) in the 2nd component, and the height (h) in the 3rd component. This implies separation between distances and angles, and a weak relationship between the values of height and the values of other parameters.

The data complexity analysis implied that the complexity of the problem was moderate, thus the following values were chosen for the hyperparameters grid-search: the number of hidden layers was [3, 5, 7, 8] and the number of neurons was [10, 20, 32, 64]. The results of training and testing of the DNN are detailed in Appendix III. For both models in all datasets that contain above 64 tuples, the meta parameter h had a consistently higher error than the other two parameters.

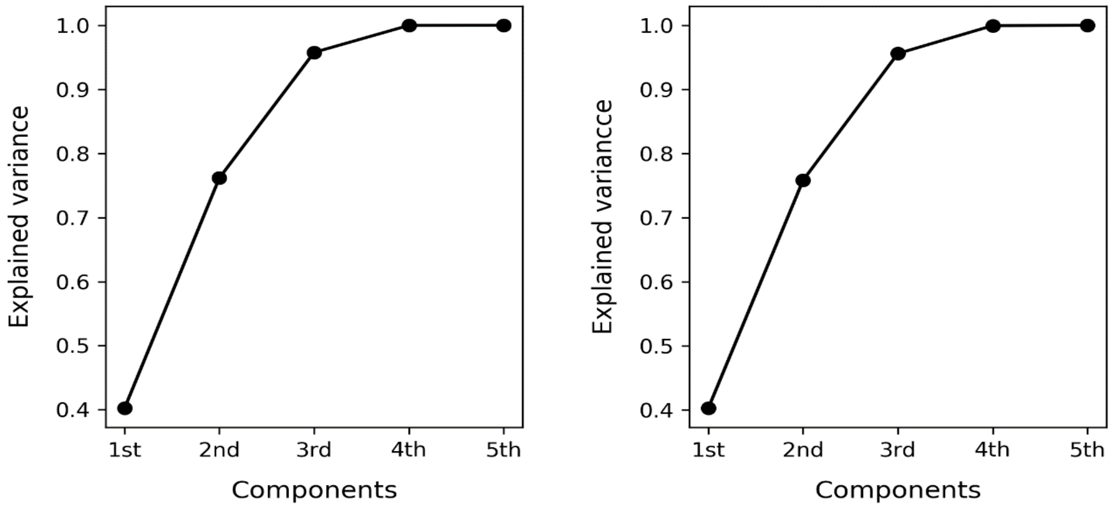


Figure 11: Parameter space analysis for the robotic soft ball throwing experiment; Cumulative variance of the principle components. Left: ballistic model; Right: adapted model (Cohen, Bar-Shira, & Berman, 2019).

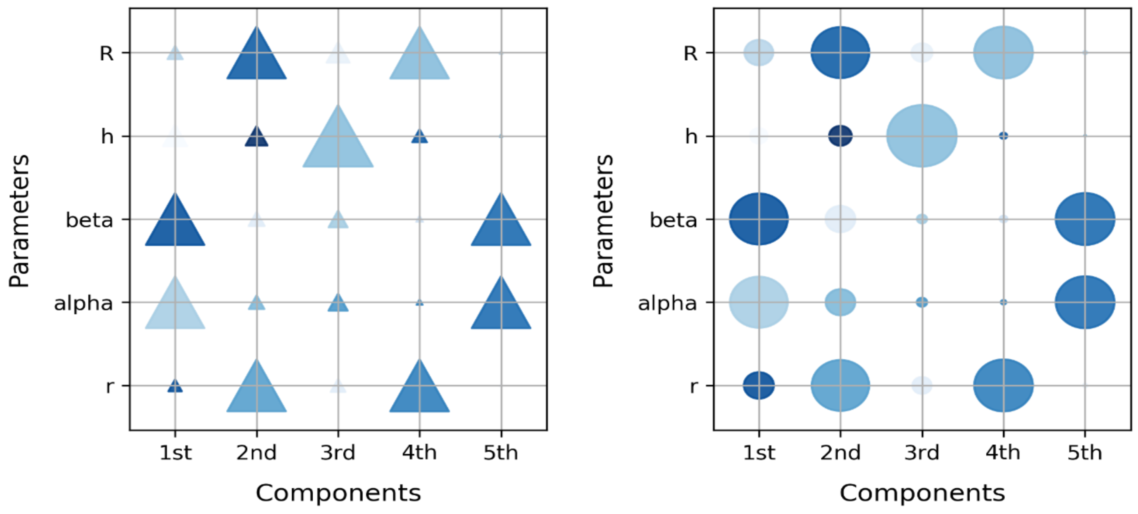


Figure 12: Parameter space analysis for the robotic soft ball throwing experiment; Weights of the parameters in the principle components. Left: ballistic model; Right: adapted model (Cohen, Bar-Shira, & Berman, 2019).

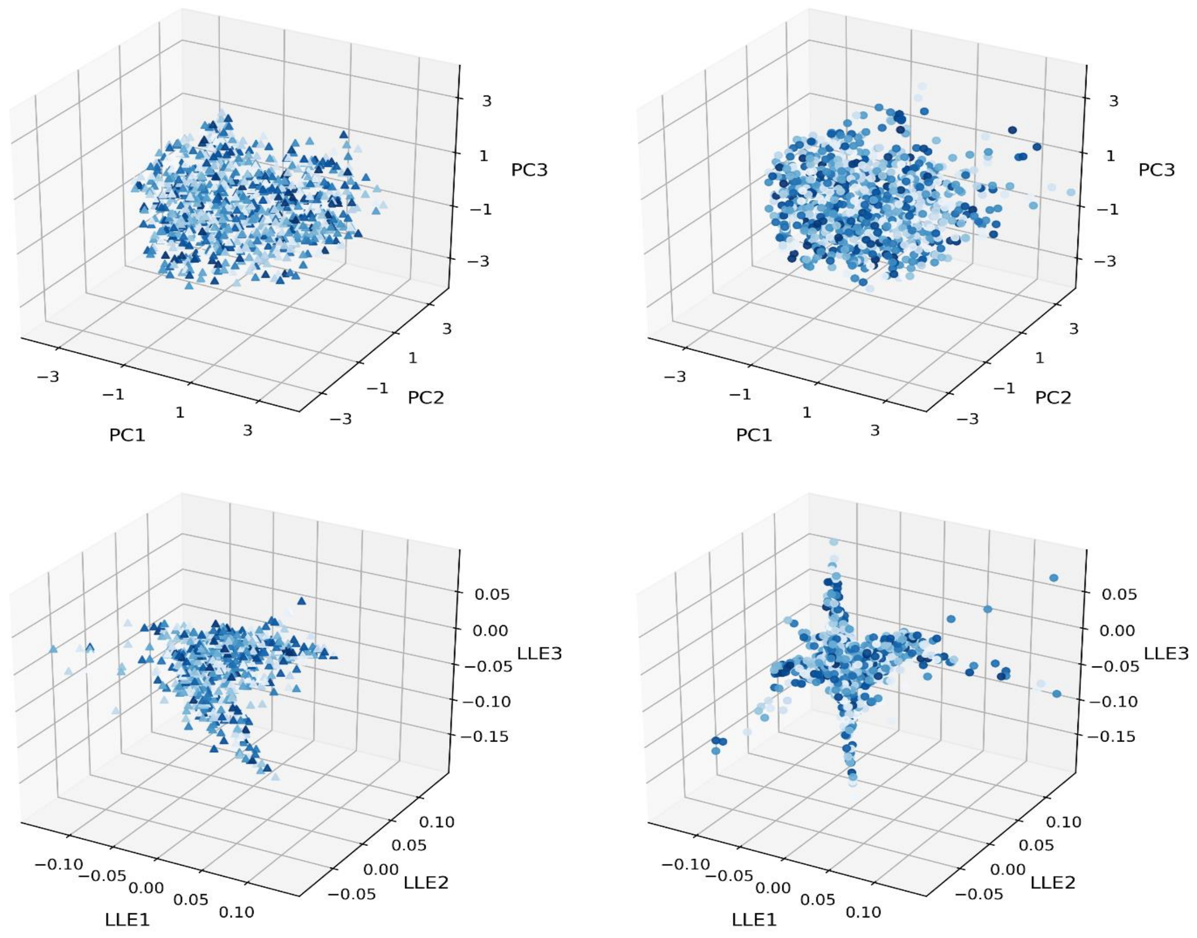


Figure 13: Parameter space analysis for the robotic soft ball throwing experiment; 3-dimantional visualization of the top 3 components (Top: PCA, Bottom: LLE). Left: ballistic model; Right: adapted model (Cohen, Bar-Shira, & Berman, 2019).

The landing point of the ball was computed based on the estimated meta parameters. For both the ballistic and adapted models and for both estimation methods tested (kernel regression and neural networks) both the distance and angle errors were smaller as dataset size increased (Figure 14). For kernel regression, the reduction in estimation error becomes very small when increasing dataset size beyond 30 tuples for the ballistic model or 64 tuples for the adapted model. For neural network, the reduction in estimation error becomes very small when increasing dataset size beyond 500 tuples for both the ballistic model and the adapted model. Beyond this value (500 tuples), the error converges in both methods to values of 3% at a distance (r) and 1° at angle (α) in the ballistic model and to values of 12% at a distance and 2° at the angle in the adapted model.

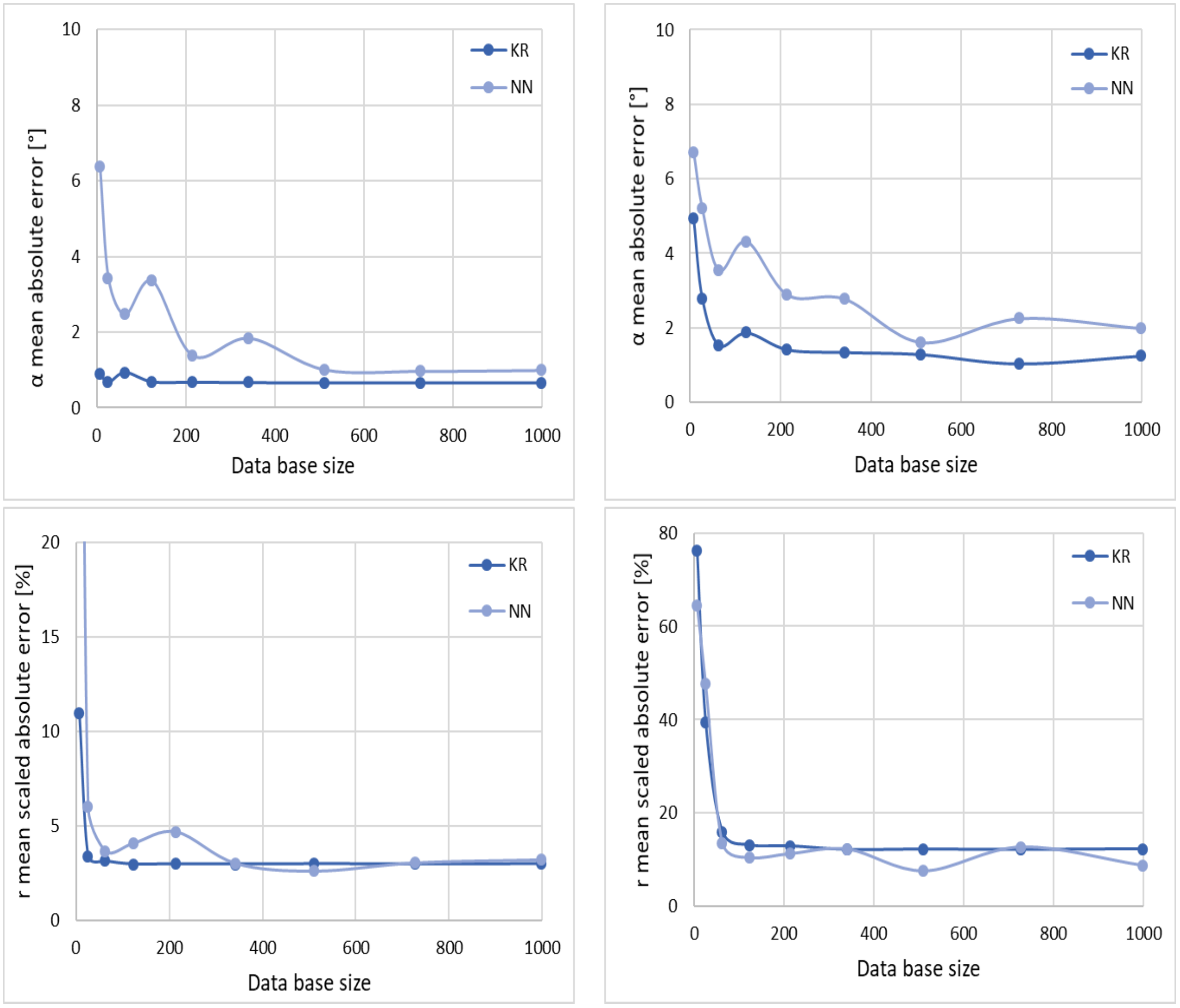


Figure 14: Movement error as a function of dataset size (Top: ball landing horizontal angle α, Bottom: ball landing radius r). Left: ballistic model; Right: adapted model (Cohen, Bar-Shira, & Berman, 2019).

6.2 Medjool thinning

PCA results showed that in order to attain above 90% of the variance all 5 components are needed and the amount of the variance explained by each PCA components is similar (Figure 15). This indicates that the complexity of the problem is relatively high. The dispersion of the first 3 PCA displays a concentration of observation that is forming a sphere-like shape and as the distance from the center of the sphere increase the dense of the observations lessen (Figure 15). The dispersion of the first 3 LLE components also displayed a concentration of observations but, unlike the PCA, when distancing from the dense area, different behavior along the different axes is observed (Figure 15).

Examining the PCA components (Figure 15) shows that the meta parameter L appears in different combination along with all task parameters. This implies a strong connection between the parameters.

The data complexity analysis implied that the complexity of the problem was high, thus the following values were chosen for the hyperparameters grid-search: the number of hidden layers was [9, 17, 21, 33] and the number of neurons was [16, 32, 64, 128].

When training the deep neural network, the combination of hyperparameters that minimized the error on the validation set was of 21 hidden layers and 128 neurons per layer. The learned mapping was tested on new task requirements and the final estimation error for the remaining length of the spikelets was 2.6%.

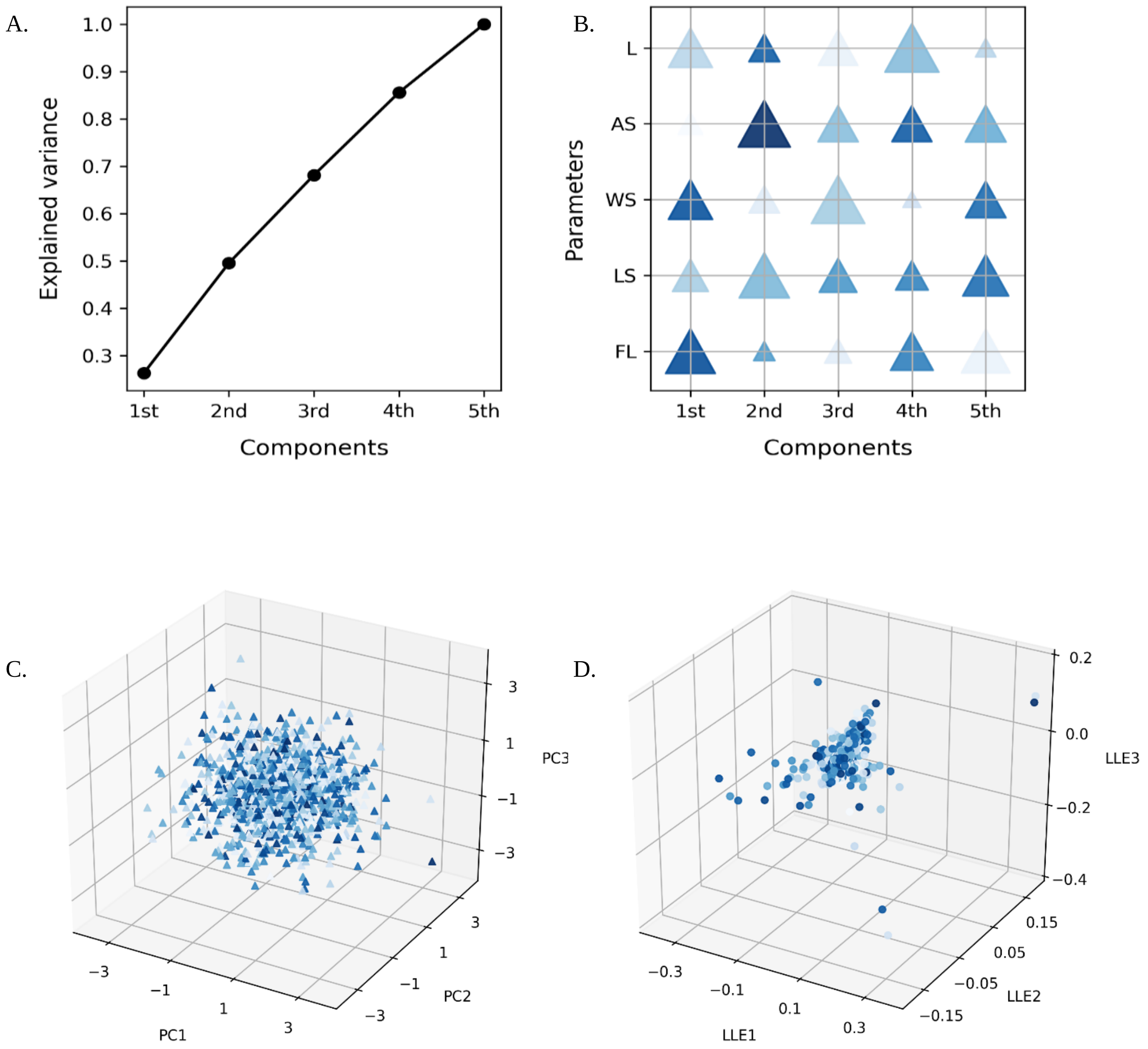


Figure 15: Parameter space analysis for the Medjool thinning experiment; A. Cumulative variance of the principle components. B. Weights of the parameters in the principle components. C. 3-dimantional visualization of the top 3 principle components. D. 3-dimantional visualization of the top 3 LLE components (Bar-Shira, et al., 2019).

Chapter 7: Discussion and future work

The research examines establishing DMP parameter values based on a mapping between task parameters and meta parameters that relies on a deep neural and a method for generating synthetic datasets to train the deep neural network.

7.1 Synthetic dataset

The integration of the stochastic fruit bunch model facilitates generation of multiple fruit bunch samples. In the last thinning season, a dataset of more than 100 physical fruit bunches was constructed. While great effort was exerted in attaining the dataset it will not be enough on its own to train the DNN. In a real environment, the cost of generating a training set with large data is expensive (long epochs, expansive equipment). Building a simulation model that takes into consideration as many physical elements as possible in order to generate a training set relaxes the difficulty of acquiring data, thus enabling the neural network to train on a large dataset which is important to enable it to reach its full competence. Moreover, the model was built in an open source environment, thus making it economically. This is an important step towards integrating robotics and state of the art machine learning algorithms to this domain.

For the fruit bunch model, integration of the model parameters with the smooth Bezier curve, allowed an ease representation of a major plant feature. Validation of additional epochs of the model is in progress. Presently, it is important to concentrate on epoch-whorl combinations that are more significant in terms of fruit quality (farmers prioritize the upper whorls), and are more convenient for the robotic operation (the structure of the fruit bunch in the early epochs is more convenient for bundling the spikelets). Additionally, collecting samples of physical fruit bunches from the chosen combinations can enable to perform goodness of fit tests for the model's distribution, hence increasing its validity and thus its acceptance in the farmers community.

The quantification of the error of the spikelets remaining length was important since it is the meta parameter of the DMP required for the bundling motion. In addition, the ground truth for each fruit bunch in the current work was established based on length (the radial length towards the 12th fruitlet). A relative error was used since the acceptable error is relative to the size of the fruit bunch.

7.2 Learning motion parameters

Analyzing the structure of the parameter manifold was instrumental in adapting movement parameters. Principal component analysis facilitated setting suitable hyperparameter search limits. Furthermore, it enables to envisage the feasibility of the DNN to successfully learn the required mapping by viewing the connections between the task and meta parameters. Absence of components that combine strong influence of a meta parameter together with at least one of the task parameters may result in low competence of the network to estimate the meta parameter. Examining the locally linear embedding, for which the linearity assumption is relaxed was important for reviewing constraints in the data and gaining a deeper understanding of model complexity. When the layout in the 3-simantional space indicates different distribution features along different axes, small error in the meta parameter estimation may lead to big movement error thus big emphasis needs to be placed on the architecture and tuning of the DNN. The analysis expedited the training of the DNN and facilitated correct emphasis to reducing errors of the more influential parameters.

Learning a mapping with small datasets is inadequate for DNNs. The parameters representing task requirement and the meta parameters have a nonlinear relationship. For the neural network to generalize, it needs to see large amount of data that covers the parameter space. When establishing big amount of data is not feasible, kernel regression may be adequate.

Dynamic movement primitives are suited for the required complex trajectories and the dynamic interactions with the environment. The motion adapted based on the mapping is predictable leading to robust task performance. For robotic applications that require complete assurances of run-time behavior without forsaking run-time adaptability such traits are critical. Moreover, the continuous mapping suites the variety that exists in dynamic environment.

In the robotic soft ball throwing experiment, the analysis of both simulations of the soft ball throwing task indicated that the complexity of the problem was moderate, i.e., that a rather small number of layers was required. The meta parameter h did not share a strong influence in the same principle components with none of the task parameters. This is in-line with the DNNs high estimation error for this parameter, regardless of the database size. For the other meta parameters, β and R, the estimation error decreased as the size of the dataset increased. The additional non-linear constrains in the adapted model are in line with the larger movement errors for this model using both methods (deep neural networks and kernel regression). The movement error decreased rapidly as the size of the training dataset increased for both kernel estimation the deep neural network. While for small datasets kernel estimation outperformed the deep neural network, for the larger datasets their performance was similar. Differently, in the Medjool thinning experiment, the analysis of the thinning task indicated that the complexity of the problem was high, i.e., that a large number of hidden layers, or a large the number of neurons in each hidden layer, or both is required. The meta parameter L shared a strong influence in the same principle component with all the task parameters. This is in-line with the small estimation error that the DNN achieved. Since the estimation error of the meta parameter was small, a small movement error is expected despite the distribution features in the LLE parameter space.

7.3 Mediation between simulation and physical environment

An important manner that requires future care is the mediation between simulation and physical environment. Additional tuning of the DNN might be needed. This can be achieved by amalgamation of this dataset with synthetic fruit bunch images integrated with background field images. It is a promising direction currently under development along with testing the learned mapping on data of physical fruit bunches.

For the current project the input for the DNN is the processed output from the sensing system, composed from both a sonar and a camera. Eliminating the pre-proccing can be achieved by constructing a network that takes as input the raw data from the sonar and camera and directly calculates the remaining length of the spikelets. The architecture of such a network needs to be carefully constructed since combining different types of inputs is not trivial. Furthermore, such network may need an increased computation ability during training and possibly in runtime as well, thus economic considerations are needed.

References

Argall, B. D., Chernova, S., Veloso, M., & Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and Autonomous Systems, 57*(5), 469–483.

Arkin, R. C. (1998). *Behavior-based robotics.* MIT Press.

Bac, C. W., Van Henten, E. J., Hemming, J., & Edan, Y. (2014). Harvesting robots for high-value crops: State-of-the-art review and challenges ahead. *Journal of Field Robotics, 31*(6), 888-911.

Bar-Shira, O., Cohen, Y., Shoshan, T., Cohen, Y., Sadowsky, A., Schmilovitch, Z., . . . Sigal, B. (2019). *Learning motion parameters for robotic Medjool date thinning.* Manuscript submitted for publication.

Barth, R., IJsselmuiden, J., Hemming, J., & Van Henten, E. J. (2018). Data synthesis methods for semantic segmentation in agriculture: A capsicum annuum dataset. *Computers and Electronics in Agriculture, 144*, 284-296.

Basri, R., & Jacobs, D. W. (2017). Efficient representation of low-dimensional manifolds using deep networks. *arXiv preprint arXiv:1602.04723.*

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research, 13*, 281-305.

Chiroma, H., Abdullahi, U. A., Abdulhamid, S. M., Alarood, A. A., Gabralla, L. A., Rana, N., . . . Herawan, T. (2018). Progress on artificial neural networks for big data analytics: A survey. *IEEE Access, 7*, 70535-70551.

Cohen, Y., & Berman, S. (2013). Tight dynamic movement primitives for complex trajectory generation. *2013 IEEE International Conference on Systems, Man, and Cybernetics* (pp. 2402-2407). IEEE.

Cohen, Y., & Glasner, B. (2015). Date palm status and perspective in Israel. In *Date palm genetic resources and utilization* (pp. 265-298). Springer.

Cohen, Y., Bar-Shira, O., & Berman, S. (2019). *Run-time adaptation based on a mapping of dynamic movement primitive parameters.* Manuscript submitted for publication.

Da Silva, I. N., Spatti, D. H., Flauzino, R. A., Liboni, L. H., & dos Reis Alves, S. F. (2017). *Artificial neural networks.* Springer.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., . . . Lempitsky, V. (2016). Domain-adversarial training of neural networks. *The Journal of Machine Learning Research, 17*(1), 2096-2030.

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. *The Fourteenth International Conference on Artificial Intelligence and Statistics* (pp. 315-323).

Hoffmann, H., Pastor, P., Park, D. H., & Schaal, S. (2009). Biologically-inspired dynamical systems for movement generation: automatic real-time goal adaptation and obstacle avoidance. *2009 IEEE International Conference on Robotics and Automation* (pp. 2587-2592). IEEE.

Hoos, H., Ca, U. B., & Leyton-Brown, K. (2014). An efficient approach for assessing hyperparameter importance. *International Conference on Machine Learning* (pp. 754-762).

Ijspeert, A. J., Nakanishi, J., & Schaal, S. (2001). Trajectory formation for imitation with nonlinear dynamical systems. *2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the Next Millennium (Cat. No. 01CH37180) (*pp. 752-757). IEEE.

Ijspeert, A. J., Nakanishi, J., & Schaal, S. (2002). Movement imitation with nonlinear dynamical systems in humanoid robots. *2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292) (*pp. 1398-1403). IEEE.

Ijspeert, A. J., Nakanishi, J., Hoffman, H., Pastor, P., & Schaal, S. (2013). Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation, 25*(2), 328-373.

Jaderberg, M., Simonyan, K., Vedaldi, A., & Zisserman, A. (2014). Synthetic data and artificial neural networks for natural scene text recognition. *arXiv preprint arXiv:1406.2227*.

Kober, J., Bagnell, J. A., & Peters, J. (2013). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research, 32*(11), 1238-1274.

Kober, J., Oztop, E., & Peters, J. (2011). Reinforcement learning to adjust robot movements to new situations. *Twenty-Second International Joint Conference on Artificial Intelligence* (pp.2650-2655).

Kober, J., Wilhelm, A., Oztop, E., & Peters, J. (2012). Reinforcement learning to adjust parametrized motor primitives to new situations. *Autonomous Robots, 33*(4), 361-379.

Lee, J. (2010). *Introduction to topological manifolds*. Springer.

Lewis, R. A. (2001). *CRC dictionary of agricultural sciences.* CRC Press.

Liu, K., Weissenfeld, A., & Ostermann, J. (2006). Parameterization of mouth images by LLE and PCA for image-based facial animation. *2006 IEEE International Conference on Acoustics Speech and Signal Processing* (pp. 461-464). IEEE.

Moustafa, A. A. (1998). Studies on fruit thinning of date palms. *First International Conference on Date Palms* (pp. 354-364).

Mülling, K., Kober, J., Kroemer, O., & Peters, J. (2013). Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research, 32*(3), 263-279.

Mussa-Ivaldi, F. A. (1999). Modular features of motor control and learning. *Current Opinion in Neurobiology, 9*(6), 713–717.

News from the grove. (2011). Retrieved from The golden date: http://www.the-golden-date.com/category/news-from-the-grove/

Pastor, P., Hoffmann, H., Asfour, T., & Schaal, S. (2009). Learning and generalization of motor skills by learning from demonstration. *2009 IEEE International Conference on Robotics and Automation* (pp. 763-768). IEEE.

Pastor, P., Kalakrishnan, M., Meier, F., Stulp, F., Buchli, J., Theodorou, E., & Schaal, S. (2013). From dynamic movement primitives to associative skill memories. *Robotics and Autonomous Systems, 61*(4), 351-361.

Potena, C., Nardi, D., & Pretto, A. (2016). Fast and accurate crop and weed identification with summarized train sets for precision agriculture. *Intelligent Autonomous Systems (IAS)* (pp. 105-121). Springer, Cham.

Prechelt, L. (1998). Early stopping-but when?. In *Neural networks: Tricks of the Trade* (pp. 55-69). Springer.

Roweis, S., & Saul, L. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science, 290*(5500), 2323-2326.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747.*

Rueckert, E., Mundo, J., Paraschos, A., Peters, J., & Neumann, G. (2015). Extracting low-dimensional control variables for movement primitives. *2015 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 1511-1518). IEEE.

Schaal, S., Mohajerian, P., & Ijspeert, A. (2007). Dynamics systems vs. optimal control-a unifying view. *Progress in Brain Research, 165*, 425–445.

Schaal, S., Peters, J., Nakanishi, J., & Ijspeert, A. (2005). Learning movement primitives. In *Robotics research. The eleventh international symposium* (pp. 561-572). Springer.

Shreiner, D., Sellers, G., Kessenich, J., & Licea-Kane, B. (2013). *OpenGL programming guide: The official guide to learning OpenGL, version 4.3.* Addison-Wesley.

Singh, N. (1995). *Systems approach to computer-integrated design and manufacturing.* John Wiley & Sons, Inc.

Smetana, L. K., & Bell, R. L. (2012). Computer simulations to support science instruction and learning: A critical review of the literature. *International Journal of Science Education, 34*(9), 1337-1370.

Stulp, F., & Schaal, S. (2011). Hierarchical reinforcement learning with movement primitives. *2011 11th IEEE-RAS International Conference on Humanoid Robots* (pp. 231-238). IEEE.

Sze, V., Chen, Y. H., Yang, T. J., & Emer, J. S. (2017). Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, *105*(12), 2295-2329.

Tamosiunaite, M., Nemec, B., Ude, A., & Worgotter, F. (2011). Learning to pour with a robot arm combining goal and shape learning for dynamic movement primitives. *Robotics and Autonomous Systems, 59*(11), 910 – 922.

Varga, T., & Bunke, H. (2003). Generation of synthetic training data for an HMM-based handwriting recognition system. *Seventh International Conference on Document Analysis and Recognition* (pp. 618-622). IEEE.

Wang, J. (2012). *Geometric structure of high-dimensional data and dimensionality reduction.* Heidelberg: Springer.

Wold, S., Esbensen, K., & Geladi, P. (1987). Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, *2*(1-3), 37-52.

Xu, R. (2015). *Machine learning for real-time demand forecasting.* Doctoral dissertation, Massachusetts Institute of Technology.

Zhang, N., Wang, M., & Wang, N. (2002). Precision agriculture—a worldwide overview. *Computers and Electronics in Agriculture, 36*(2-3), 113-132.

Zhang, Y., Guo, Q., & Wang, J. (2017). Big data analysis using neural networks. *10.15961/j.jsuese.2017.01.002, 49*, 9-18.

Zhou, J., & Zhang, B. (Eds.). (2019). *Agricultural robots: Fundamentals and applications.* BoD–Books on Demand.

האוסלר [Hausler], ד., פרידקין [Fridkin], צ., & קחל [Kachel], י. (2017). *ענף התמרים*. מדינת ישראל, משרד החקלאות ופיתוח הכפר, החטיבה למחקרת כלכלה ואסטרטגיה.

כהן [Cohen], י'. (2014). ייחודו של עץ התמר והפיתוחים הטכנולוגיים הדרושים לו. *ניר ותלם- ירחון לנושאי גידולי שדה מיכון והנדסה בחקלאות*, 45-46.

מועצת הצמחים [The plant council]. (2019). מפקד מטעי התמרים.

Appendix I: Medjool date fruit bunch model

Table 2 present the full model devised for capturing the geometry of the Medjool date fruit bunch. Eighteen parameters were defined. A probability function was defined for each of the parameters and distribution parameters were fitted for the three thinning epochs and the three whorls (9 sets of probability parameters). Final validation was performed for the parameters of fruit bunches from the 1st epoch-1st whorl.

Table 2: Medjool date fruit bunch full model parameter distributions.

Index	**Parameter**	**Probability function**[a]	**Whorl**	**Epoch**		
				1st	**2nd**	**3rd**
p_1	Rachis width [cm]	$U(a,b)$	1st	$(2,3)$	$(2,3)$	$(3,4)$
			2nd	$(3,4)$	$(3,4)$	$(4,5)$
			3rd	$(4,5)$	$(4,5)$	$(5,6)$
p_2	Rachis depth [cm]	$U(a,b) * (p_1)$	1st	$(0.56,0.76) \times (p_1)$		
			2nd			
			3rd			
p_3	Viewed section of rachis [cm]	$Constant$	1st	20		
			2nd			
			3rd			
p_4	Hidden Rachis length [cm]	$Triang(a,b,c)$	1st	$(20, 25, 30)$	$(22,25,32)$	$(29,35,40)$
			2nd	$(17,19,25)$	$(15,23,30)$	$(25,32,37)$
			3rd	$(10,13,15)$	$(14,20,24)$	$(22,25,30)$
p_5	Angle between rachis to tree trunk [°]	$U(a,b)$	1st	$(0, 10)$	$(10,20)$	$(15,30)$
			2nd	$(10,17)$	$(15,25)$	$(45,60)$
			3rd	$(10,20)$	$(30,40)$	$(60,90)$
p_6	Number of spikelets	$Triang(a,b,c)$	1st	$(80,85,100)$		
			2nd	$(50,70,80)$		
			3rd	$(50,60,65)$		
p_7	Number of spikelets clusters	$Constant$	1st	3		
			2nd			
			3rd			
p_8	Distance of spikelets clusters from		1st	$Bottom\ cluster$: $0 \times (p_4)$ $Middle\ cluster$: $0.4 \times (p_4)$ $Top\ cluster$: $0.6 \times (p_4)$		
			2nd			

	top of presented rachis		3rd			
p_9	Dispersion of spikelet quantity between the clusters		1st	$Bottom\ cluster$: $0.5 \times (p_6)$ $Middle\ cluster$: $0.3 \times (p_6)$ $Top\ cluster$: $0.2 \times (p_6)$		
			2nd			
			3rd			
p_{10}	Spikelet length [cm]	$Triang(a,b,c)$	1st	$(75, 80, 100)$		
			2nd	$(65,79,89)$		
			3rd	$(50,60,65)$		
p_{11}	Spikelet radius [cm]	$U(a,b)$	1st	$(2,2.5)$		
			2nd			
			3rd			
p_{12}	Clamps width (i.e. the largest distance between two spikelets) [cm]	$Triang(a,b,c)$	1st	$(25, 27, 35)$	$(35,40,45)$	$(70,80,100)$
			2nd	$(20,24,30)$	$(30,35,40)$	$(50,70,90)$
			3rd	$(20,21,24)$	$(24,30,35)$	$(50,70,85)$
p_{13}	Number of fruitlets per spikelet (on first cluster)	$U(a,b)$	1st	$(60,70)$	$(40,50)$	$(25,35)$
			2nd	$(50,60)$	$(30,40)$	$(17,25)$
			3rd	$(35,45)$	$(25,35)$	$(12,22)$
p_{14}	Distance from the spikelet base to the first fruitlet [cm]	$Triang(a,b,c)$	1st	$(20,27,45)$		
			2nd			
			3rd			
p_{15}	Fruitlet radius [cm]	$Triang(a,b,c)$	1st	$(1, 1.2, 1.5)$	$(1.5,1.8,2)$	$(1.8,1.8,2.5)$
			2nd	$(0.9,1.1,1.2)$	$(1,1.2,1.5)$	$(1.5,1.8,2)$
			3rd	$(0.7,0.8,0.9)$	$(0.9,1.1,1.2)$	$(1,1.2,1.5)$
p_{16}	Distance between two adjacent fruitlets [cm]	$Exp(\frac{1}{\lambda})$	1st	(1.3)		
			2nd			
			3rd			
p_{17}	Natural fallout probability [%]	$Berr(p)$	1st	16	30	60
			2nd			
			3rd			
p_{18}	Bending of the fruit bunch [%]	$Constant$	1st	0	0.2	0.2
			2nd			
			3rd			

a. U(a,b) – uniform distribution; Triang(a,b,c) – triangular distribution; Berr(p) – Bernoulli distribution; Exp($\frac{1}{\lambda}$) – exponential distribution.

Appendix II: Code description

Fruit bunch dataset generation

Generates a dataset of images and accompanying properties of Medjool date fruit bunches. To initial a new experiment, the user enters an experiment number, seed, dataset size, and the required whorl and epoch. Furthermore, the user can choose whether to draw the cutting plain. The code automatically opens a new directory to which images and a csv file containing related information are saved. The code was developed with python 3.7. The libraries used are detailed in Table 3 and the main methods are detailed in Table 4.

Table 3: Libraries used in the 'Fruit bunch dataset generation' code.

Library name	**Description**
pygame	Offers computer graphics abilities. It is a python wrapper module for the SDL multimedia library.
OpenGL	Offers rendering abilities for 2D and 3D vector graphics.
math	Offers access to the mathematical functions defined by the C standard.
numpy	Offers scientific computing tools.

Table 4: Main methods in the 'Fruit bunch dataset generation' code.

Method name	**Description**
init_exp	Sets model parameters according to the user's input regarding the required whorl and epoch.
compute_bezier_points	Computes the points along a curve, given four Bezier control points.
draw_sphere	Draws a sphere, given the required center and radius.
draw_bezier_points	Draws a curve, given a set of points that are calculated by the "compute_bezier_points" method.
draw_elliptical_cylinder	Draws an elliptical cylinder, given its girth, height, and center of its lower base.
draw_cone	Draws a cone, given the girth and center of its base, and its height.

draw_spikelets	Computes four control points for each spikelet and activates the "draw_bezier_points" method.
take_screen_shoot	Saves an image of the fruit bunch to the experiment's directory.
main	Runs the main loop that activates pyGame and creates the fruit bunch 3d graphics by adjusting the lighting and calling the methods above.
Generate dataset	Receives user's input and generate a dataset by calling the 'main' method and saving the data to the experiment's directory.

Complexity analysis

Performs PCA and LLE. To initial a new experiment, the user uploads the required dataset of related task and meta parameters. Then, data preparation is conducted, both methods are applied on the dataset, and figures are displayed and saved to a directory defined by the user. The code was developed with python in google colaboratory. The libraries used are detailed in Table 5 and the flow of the code is detailed in Table 6.

Table 5: Libraries used in the 'complexity analysis' code.

Library name	**Description**
pathlib	Offers a set of classes to handle filesystem paths.
matplotlib.pyplot	Offers a MATLAB-like plotting framework.
pandas	Offers data analysis / manipulation tool.
numpy	Offers scientific computing tools.
random	Offers pseudo-random number generators for various distributions.
seaborn	Offers statistical graphics tools. It is built on top of matplotlib and closely integrated with pandas data structures.
sklearn	Offers machine learning tools.
mpl_toolkits.mplot3d	Offers an extended 3d plotting abilities.

Table 6: Flow of the ‘complexity analysis’ code.

Section name	**Description**
Load data	Loads dataset to the colaboratory.
Data preparation	Normalizes the dataset so that the values are between zero and one.
PCA	Applies PCA on the data.
LLE	Applies LLE on the data.
Figures	Displays and saves the following images: cumulative variance of the principle components, weights of the parameters in the principle components, 3d visualization of the top 3 principle components, and 3d visualization of the top 3 LLE components

DNN: train and test

Trains and tests a deep neural network. To initial a new experiment, the user uploads the required dataset of related task and meta parameters (it is optional to upload one dataset that will later be divided to a training set and a test set, or to upload both training set and test set). Then, data preparation is conducted, the DNN is trained and the values for the hyperparameters are chosen, the final model is tested on the test set, and the estimation errors of the meta parameters are calculated. There is an option to build and export a dataset containing the task parameters and the predicted meta parameters. The code was developed with python in google colaboratory. The libraries used are detailed in Table 7 and the flow of the code is detailed in Table 8.

Table 7: Libraries used in the 'DNN: train and test' code.

Library name	**Description**
tensorflow	Offers high performance numerical computation abilities.
pathlib	Offers a set of classes to handle filesystem paths.
matplotlib.pyplot	Offers MATLAB-like plotting framework.
pandas	Offers data analysis / manipulation tool.

numpy	Offers scientific computing tools.
random	Offers pseudo-random number generators for various distributions.
seaborn	Offers statistical graphics tools. It is built on top of matplotlib and closely integrated with pandas data structures.

Table 8: Flow of the 'DNN: train and test' code.

Section name	**Description**
Fix seed	Control all pseudo-random number generators to enable an experiment to be repeated.
Load data	Loads dataset to the colaboratory.
Data preparation	Divides the dataset to training set and test set if needed; Creates four datasets (training set, training labels, test set, and test labels) by separating the labels from the full training set and the full test set; Normalizes the training set and test set so that the values are between zero and one (the normalization is performed using the training set characteristics).
Hyperparameters tuning	Performs a grid-search to adjust the number of hidden layers, the number of neurons in each layer, and the learning rate of the optimizer.
Build model	Trains the DNN with the chosen architecture.
Test model	Tests the trained DNN on the test set and computes estimation errors of the meta parameters.
Generate dataset	builds and exports a dataset containing the task parameters and the predicted meta parameters.

Appendix III: DNNs training and testing results for the robotic soft ball throwing experiment

Table 3 present the results of training and testing of the DNNs in the robotic soft ball throwing experiment. A DNN was trained for each of the eighteen training sets, nine training set for each model (ballistic and adapted). the results of the grid-search that was performed for three of the DNN hyperparameters (learning rate of the optimizer, number of hidden layers, and number of neurons in each layer) are detailed. The trained DNNs were testes on two test sets, one for each model, and the results of the meta parameters estimation errors are presented. All datasets are available in Mendeley Data (https://data.mendeley.com/datasets/d75vv3tdkg/1)

Table 9: DNNs training and testing results for the robotic ball throwing experiment.

Model	**Training set size**	**Hyperparameters optimization results**			**Meta parameters estimation error**		
		Learning rate	Hidden layers	Neurons	MAE_{β} [°]	$MSAE_{h}$ [%]	$MSAE_{R}$ [%]
Ballistic	8	0.001	3	64	2.964	0.294	0.739
Ballistic	27	0.001	8	64	2.076	0.281	0.026
Ballistic	64	0.0005	8	32	1.48	0.315	0.015
Ballistic	125	0.001	8	64	2.117	0.292	0.017
Ballistic	216	0.0005	7	64	0.8	0.291	0.01
Ballistic	343	0.0005	7	32	1.043	0.291	0.007
Ballistic	512	0.0005	7	20	0.585	0.3	0.005
Ballistic	729	0.001	7	32	0.496	0.294	0.004
Ballistic	1000	0.0005	7	64	0.645	0.294	0.004
Adapted	8	0.001	3	64	2.561	0.325	0.8
Adapted	27	0.001	3	64	2.513	0.33	0.788
Adapted	64	0.0005	7	64	1.798	0.286	0.027
Adapted	125	0.0005	8	32	2.704	0.28	0.019
Adapted	216	0.0005	7	64	1.592	0.285	0.013
Adapted	343	0.0005	8	64	1.514	0.282	0.01
Adapted	512	0.001	7	32	0.81	0.294	0.009
Adapted	729	0.0005	7	64	1.214	0.286	0.01
Adapted	1000	0.0005	7	32	1.018	0.285	0.006

תקציר

דילול חנטים הינה משימה חשובה בתהליך הגידול של תמר המגה'ול. דילול יעיל ומתוזמן הינו הכרחי על מנת להשיג איכות פרי גבוהה. כיום, דילול דורש דיוק, כוח אדם גדול והוא מוגבל בזמן. דילול של עץ בודד מוערך בכ-3.5 שעות ומעל למיליון שעות עבודה דרושות על מנת לדלל את כל עצי המגה'ול הקיימים כיום בישראל. אוטומציה של פעולת הדילול הינה הכרחית בכדי לאפשר דילול של מספר רב של מטעים עם פחות כוח אדם ובמסגרת הזמן הרצויה.

בקרת תנועה במהלך ביצוע הדילול הינה מאתגרת כיוון שנדרשת תנועה אל האשכולות בתוך סבך העץ. בנוסף, הסביבה הינה דינאמית ולכן יכולת להסתגל לשינויים במהלך זמן הריצה היא הכרחית. אנחנו מציגים שיטה לתכנון ובקרת התנועה עבור מערכת רובוטית לצורך דילול של תמר מג'הול. השיטה מבוססת על תבניות תנועה דינאמיות (dynamic movement primitives). לפי שיטה זו, קידוד התנועה של המערכות הדינאמיות נעשה בעזרת סט משוואות לא לינאריות אשר להן פרמטרים שערכיהם ניתנים לכוונון על מנת לאפשר ביצוע של משימות חדשות. הפרמטרים קבועים את הדיוק של התנועה המיוצרת וערכיהם מושפעים מהמשימה, הרובוט והסביבה. את הפרמטרים ניתן לחלק לשלוש קטגוריות: פרמטרי צורה (מגדירים את צורת התנועה במרחב), פרמטרי על (מגדירים את מרחב הפעולה. לדוגמא, נקודת התחלה ונקודת יעד) ופרמטרים חיצוניים (מגדירים אילוצי סביבה חיצוניים. לדוגמא, השהייה בתחילת התנועה).

אנחנו מציגים דרך לבצע אדפטציה של הפרמטרים המבוססת על ידי יצירת מיפוי בין הפרמטרים המגדירים את המשימה אל פרמטרי העל. המיפוי מאפשר יכולת הכללה ודיוק בעת יצירת תנועה עבור משימה חדשה, תכונות חשובות עבור פעולת הדילול. בנוסף, המיפוי נלמד מראש דבר אשר מבטיח את אמינותו במהלך זמן הריצה. מכיוון שלמידת המיפוי היא בעיה בעלת קלטים ופלטים מרובים, ניתן לבצע את הלמידה באמצעות רשת נוירונים עמוקה (deep neural network), בה פרמטרי המשימה מהווים את שכבת הקלט ופרמטרי המטה הם שכבת הפלט. אימון הרשת הואץ על ידי ניתוח מרחב הפרמטרים באמצעות ניתוח גורמים ראשיים (principal components analysis) והטמעה לינארית מקומית (locally linear embedding).

כדי להשיג דיוק טוב, הרשת צריכה להתאמן על בסיס נתונים גדול. מחסור בבסיסי נתונים חקלאיים נחשב בין הסיבות העיקריות לביצועים נמוכים יותר של אלגוריתמי למידת מכונה בחקלאות ובין צווארי הבקבוק העיקריים המעכבים כניסה של רובוטיקה לחקלאות. כיוון שבנית בסיס נתונים המכיל מדידות פיזיקליות עבור משימת הדילול הינה מאתגרת, פותחה שיטה לבניית בסיס נתונים המכיל נתונים שיוצרו באופן מלאכותי. נבנה מודל הסתברותי המתאר אשכול של תמר מג'הול ובוצעה לו וויזואליזציה תלת ממדית בעזרת הספרייה הגראפית של הקוד הפתוח בפיתון (python openGL). האשכול מודל כאוסף של צורות גיאומטריות, ולכל צורה הותאמה התפלגות ייחודית.

היכולת לייצר תנועה מדויקת באמצעות האדפטציה לשיטת תבניות תנועה דינאמיות נבדקה בניסוי מקדים עבור משימה של זריקת כדור על ידי זרוע רובוטית שמודלה בסימולציה. עבור משימה זו, פרמטרי המשימה מתארים את נקודת הנחיתה של הכדור על הרצפה. שני מודלים של זריקת כדור נבנו בסימולציה, מודל בליסטי ומודל מותאם. בנוסף, השימוש ברשת הושווה לשימוש ברגרסיה קרנלית (kernel regression). ניתוח מרחב הפרמטרים הצביע על כך שמורכבותם

של שני המודלים, הבליסטי והמותאם, הייתה בינונית עם אילוצים חזקים יותר במודל המותאם. עבור בסיסי נתונים גדולים, שגיאות התנועה שהשיגו שתי שיטות המיפוי היו נמוכות (שגיאת מרחק הקטנה מ-12% ושגיאת זווית הקטנה מ-2 מעלות). ניסוי שני בדק את יכולת הרשת ללמוד את המיפוי בין פרמטרי המשימה לפרמטרי העל עבור משימת הדילול תוך שימוש בסיסי נתונים שיוצרו באמצעות מודל האשכול. ניתוח מרחב הפרמטרים הצביע על כך שמורכבות הבעיה הייתה גבוהה. לאחר האימון, הרשת השיגה שגיאה חיזוי נמוכה מאוד של 2.6%. התוצאות שהתקבלו הינן מבטיחות, והשיטה והכלים שפותחו מוכנים לבדיקה בניסוי שטח.

מילות מפתח: תבניות תנועה דינאמיות, רובוטיקה בחקלאות, רשתות עמוקות, נתונים מלאכותיים.

אוניברסיטת בן-גוריון בנגב
הפקולטה למדעי ההנדסה
המחלקה להנדסת תעשייה וניהול

לימוד פרמטרי תבניות תנועה עבור רובוט לדילול תמר מג׳הול

חיבור זה מהווה חלק מהדרישות לקבלת תואר מגיסטר בהנדסה

מאת : אור בר-שירה

מנחה : פרופ׳ סיגל ברמן

חתימת המחבר : אור בר-שירה Or .... תאריך : 28/09/2019

אישור המנחה : סיגל ברמן סיגל ברמן .... תאריך : 28/09/2019

אישור יו״ר ועדת תואר שני מחלקתית : [illegible] .... תאריך : 28/9/2019

אוניברסיטת בן-גוריון בנגב

הפקולטה למדעי ההנדסה

המחלקה להנדסת תעשייה וניהול

לימוד פרמטרי תבניות תנועה עבור רובוט לדילול תמר מג׳הול

חיבור זה מהווה חלק מהדרישות לקבלת תואר מגיסטר בהנדסה

מאת: אור בר-שירה

אלול ה'תשע"ט ספטמבר 2019