DEVELOPMENT OF A TABLE-TOP ROBOT MODEL FOR THINNING OF FRUIT

BY

FU OU YANG

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Agricultural and Biological Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2012

Urbana, Illinois

ABSTRACT

Specialty crop production contributes significant revenue to agriculture in the United States. To reduce the cost of human labor, a concept of automatic selective thinning was pursued to improve the production of peach, apple and other fruits. In this research, a table-top robot model was developed to prove the concept, with an emphasis on a task planning algorithm. The algorithm moved a custom six-DOF (degree of freedom) manipulator and placed an end-effector at target blossom locations with a feasible posture.

A “floating z-axis” method was developed to place the end-effector at targets, and was successfully verified by a three-dimensional artificial branch. This method automatically searched the solution domain and selected a feasible posture using the concept of “minimum effort,” which was judged by the movements from a current posture to potential subsequent postures. To ensure a straight cut-in path from a standby posture to the thinning posture, 20 intermediate postures were assigned automatically along the proceeding direction of the end-effector. The calculation time for single target was approximately 7.7 seconds.

Peach was taken as an example in this study and a model tree was built for the verification experiments. Important issues of future development and deployment were identified, including the requirements of all subsystems, adaption of orchard systems, and other technology improvements.

To Father, Mother, and Chi-Ju

ACKNOWLEDGEMENTS

First, I am grateful to have Dr. Tony Grift as my adviser, who was fun to work with and inspired me greatly. His encouragement of trying various ideas and tolerance of making mistakes allowed me learn to a lot on the road. This research would not have been possible without his support and guidance. I have also benefited from the project participants at Pennsylvania State University. Dr. Paul Heinemann, David Lyons, Robin Pritz, and other professionals helped this research significantly. The collaboration among us was important and precious. I would also like to thank my committee members, Dr. K.C. Ting and Dr. Timothy Bretl, for providing valuable suggestions to improve this thesis. I wish I could have more discussions and interactions with them so I could accomplish more in this study.

Looking back at the journey, I am thankful to have opportunity for pursuing a master's degree in the U.S. It is not only about the opportunity to study with exceptional researchers from different places of the world, but also about the experience of living in a diverse and open-minded society. The cultural impact was tremendous and hard to measure. I truly appreciate all the people I met here, and treasure every wonderful moment we shared. The two-year stay in this country is an unforgettable chapter in my life.

TABLE OF CONTENTS

LIST OF ACRONYMS

CASC	Comprehensive Automation for Specialty Crops
DOF	Degree of Freedom
IKS	Inverse Kinematics Solution
ITTF	Innovative Technologies for Thinning of Fruit
KAC-V	Kearney Agricultural Center-V shape trees
PSU	Pennsylvania State University
SCRI	Specialty Crop Research Initiative
UCD	University of California, Davis
UIUC	University of Illinois at Urbana-Champaign
UMD	University of Maryland
USDA	United States Department of Agriculture

1 INTRODUCTION

Fruit production is an import component of agriculture in the United States, with a total production value reaching \$ 15 billion in 2010, where apple contributed \$ 2.2 billion and peach \$ 0.6 billion (USDA-NASS, 2012). These "specialty" crops require pruning and thinning during cultivation, rendering human labor as one of the major costs of production. Hence, reducing the personnel expense is a major concern for developing new horticultural technologies. To accomplish this, the concept of robotic selective thinning was pursued. The selective thinning system would consist of 1) a machine vision system that automatically generates three dimensional blossom locations, 2) a robotic arm equipped with a dedicated end-effector, and 3) a carrying platform that transports the robotic system and places it at a desired location with respect to a peach tree. This thesis describes the development of a scaled-down model robotic arm, including the hardware and software needed to place the end-effector at preset blossom locations for removal.

1.1 Research Motivations

Peach (*Prunus persica* (L.) Batsch), is an important crop in the US with a production of about 1 million Mg with a value of \$ 596 million, requiring a production area of 45,300 hectares (112,000 acres). The peach industry is the fourth most valuable non-citrus fruit business after grape, apple, and strawberry in the US (USDA-NASS, 2012).

During its cultivation period, blossom thinning is a crucial step for crop load management, since an excessive number of blossoms eventually yield too many non-salable undersized fruits and consequently, decrease revenue for peach growers. The blossoms or fruitlets must be removed to adjust the crop load and to ensure that the harvested fruits (Figure 1.1) are of commercial size (Schupp et al., 2008). Traditionally, the thinning work is performed by skilled labor using a tedious manual picking-off method. Some alternatives have been adopted with unsatisfying results; thus, a more advanced solution is desirable to address this issue.

Figure 1.1 Harvested Saturn peaches.

The USDA has sponsored a project titled "Innovative Technologies for Thinning of Fruit (ITTF) through the Specialty Crop Research Initiative (SCRI) program. One component of this program is "selective thinning" in which a quasi-anthropomorphic method based on a robotic manipulator integrated with a computer vision system was proposed for removal of blossoms.

The methodology of the robotic thinning system is illustrated by a flowchart, as shown in Figure 1.2. Four academic institutions are involved in this project: the Pennsylvania State University (PSU), the University of Illinois at Urbana-Champaign (UIUC), the University of California at Davis (UCD), and the University of Maryland (UMD). PSU focused on the development of hardware such as a custom manipulator and an end-effector. UIUC endeavored to develop a task planning algorithm to equip the robotic system with intelligence for decision making. They were also responsible for the corresponding control software to activate the robotic system. UCD and UMD developed machine vision systems to enable robotic perception.

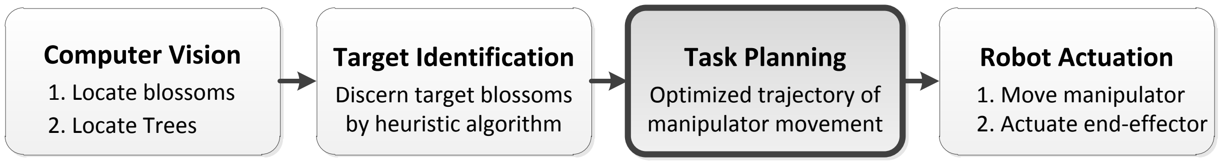
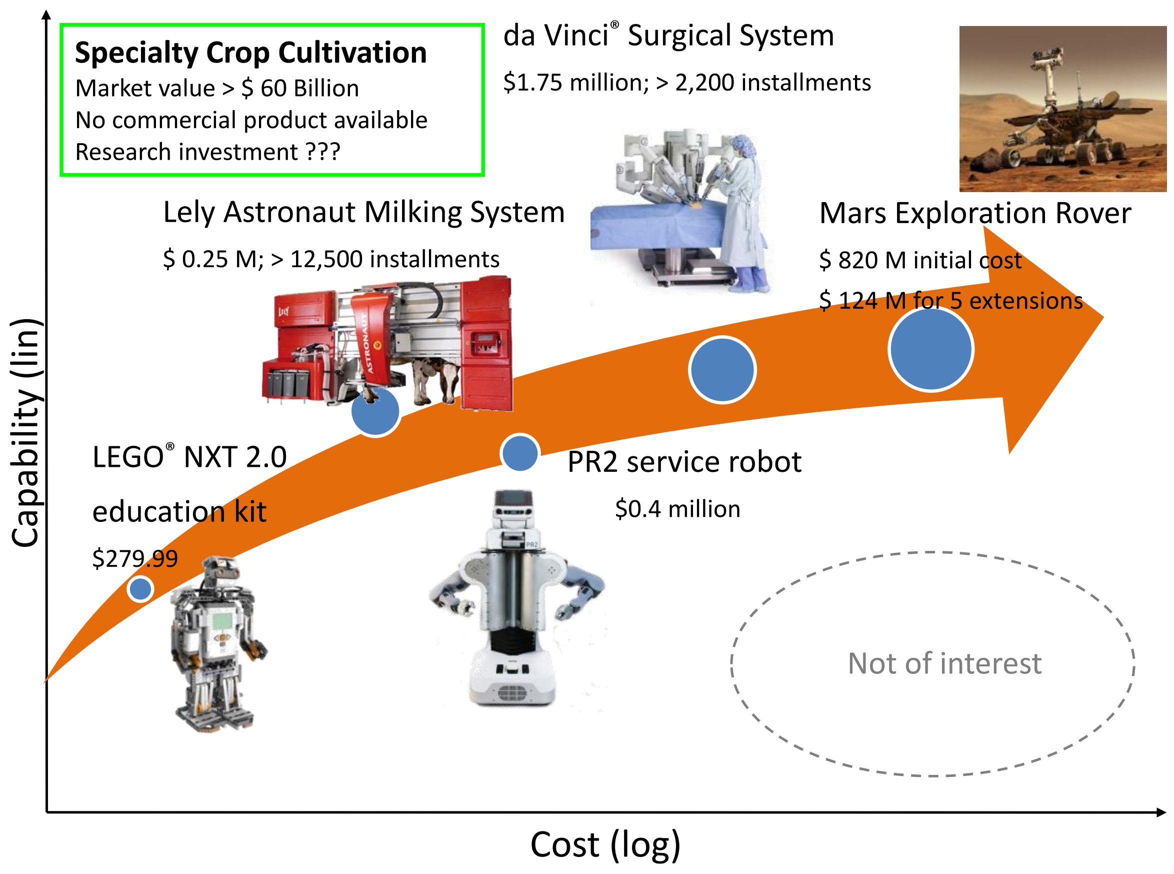


Figure 1.2 Methodology of robotic thinning system.

Commercialized agricultural robots are scarce and their impact on agriculture has been limited. Some robots from various fields are listed in Figure 1.3 where the capability of the robots is plotted versus their costs. Robots range in cost from low-cost educational kits to the most expensive robots ever conceived such as the Mars rovers. There are efforts to produce near humanoid robots such as Honda’s Asimo and the PR2 robots, which are high in cost. A stark contrast exists with agriculture, in which many prototype robots have been produced ranging from sheep shearing and automatically guided vehicles to harvesting of various fruits and vegetables. However, the only robot in agriculture that has shown significant commercial success is the milking robot. There is a clear need for robotics in agriculture where the technology could replace tedious (and often undocumented) labor, and simultaneously enhance the life of farmers and animals such as in the milking robot case. Although the interest in agricultural robotics has increased, more work is needed to bring concepts to commercialization. The desired capabilities of robots in agriculture, such as in this thinning project, are high since target objects are highly variable (trees, crops, blossoms, fruits and vegetables), in addition to having to deal with external influences such as sunlight changes, soil conditions, and wind effects. However, the cost of the robots must remain low to make them affordable for farmers and growers. This puts agricultural robotics in uncharted terrain, and the hope is that this thesis research may contribute to increasing investment in this important and challenging application domain.



Specialty Crop Cultivation
Market value > $ 60 Billion
No commercial product available
Research investment ???
da Vinci® Surgical System
$1.75 million; > 2,200 installments
Lely Astronaut Milking System
$ 0.25 M; > 12,500 installments
Mars Exploration Rover
$ 820 M initial cost
$ 124 M for 5 extensions
LEGO® NXT 2.0
education kit
$279.99
PR2 service robot
$0.4 million
Not of interest
Capability (lin)
Cost (log)

Figure 1.3 Schematic of cost-capability relationship of robots in various application domains.

1.2 Objectives

The goal of this project, which is applying a robotic thinning system in an orchard, is ambitious and needs in-depth research into the four sections as shown in Figure 1.2, followed by component integration. Instead of starting with inherently varying trees, an abstract scaled-down tree model was constructed to prove the concept of robotic thinning and to identify the most important issues to be addressed. Within this framework, the overall goal of this study was to investigate the potential of the use of robotics in thinning of fruit trees. The work was comprised of the following three objectives.

(1) Construction of a model tree representing peach tree.

(2) Development of a dedicated table-top manipulator.

(3) Development and implementation of algorithms allowing positioning of an end-effector for the removal of blossoms.

2 LITERATURE REVIEW

To illustrate both classical and modern methods used for thinning, section 2.1 discusses the merits and limitations of the methods as developed to date. Section 2.2 discusses the definition of the term robot, since over time various definitions have evolved. In section 2.3, the proliferation and achievement of robotics in agriculture is outlined, with an emphasis on manipulators in plant production. Finally, section 2.4 is devoted to the introduction of a task planning algorithm for manipulators.

2.1 Technologies of Fruit Thinning

Blossom thinning of peach trees, known as the "June drop," occurs naturally for some genotypes, such as nectarines (Bassi and Monet, 2008). This is however not the case in general, and the thinning result is not always satisfying. Typically, thinning is accomplished by human labor, chemicals, or mechanical devices (Marini and Reighard, 2008). Hand thinning by human labor is the traditional and primary approach; nevertheless, it is expensive for peach growers owing to the dwindling experienced labor force in agriculture. Glozer and Hasey (2006) showed that hand thinning labor represented 31% of all cultural costs of extra-early cling peach varieties in California.

Chemicals that kill flowers are also in use: for example, Fallahi et al. (2006) applied various chemicals on apples, peaches, nectarines and plums in varying rates to optimize the fruit set, fruit quality, and yield. Although chemical methods are effective, efficient and relatively inexpensive, the consistency of the results is poor and depends on environmental factors. Therefore, the operation is followed by hand-thinning to optimize the result (Bassi and Monet, 2008).

Ample effort has been devoted to mechanical (or physical) thinning. In recent years, the most promising solutions are found in devices such as the Darwin string thinner (Figure 2.1) manufactured by Fruit-Tec, Deggenhausertal, Germany. The Darwin machine is a general-purpose thinner with an adjustable number of cords attached to a rotating spindle, rendering its action akin to a rather crude “whipping off” of the blossoms on the trees. To optimize the performance, the peach trees should be trained to a vase shape (Figure 2.1) or a KAC-V shape. KAC-V is also known as perpendicular-V and proposed by University of California’s Kearney Agricultural Center (KAC). Some scientists and engineers have researched optimal parameters for the Darwin machines under varying experimental arrangements (Schupp et al., 2008). Although the results of mechanical devices were improved, some issues remained unsolved. The current mechanical devices are non-selective, which means that the device can only remove a certain percentage of blossoms without differentiating among them. In practice, during hand thinning, farmers adjust the spacings among blossoms and retain the ones in a favorable location with respect to the branch. In addition, the cord-based machines can damage other blossoms without removing them and potentially harm the tree bark, which adds uncertainty to the yield. Another limitation compared the proposed selective thinning using robots, is that the Darwin machine has only a single purpose, whereas a robot may be used for a variety of tasks such as branch pruning, fruitlet thinning, and fruit harvesting.

Apart from the ITTF, another USDA-sponsored project, Comprehensive Automation for Specialty Crops (CASC), focused on the development of sensing and automation technologies for orchard environments. The tasks were mainly conducted by researchers from Carnegie

Mellon University and categorized into three main themes summarized in Figure 2.2. Their GPS-free autonomous guiding vehicle (Figure 2.3) replaces the driver and removes the requirement of ascending and descending a ladder. It may serve as a carrier vehicle for the selective thinning system developed by the ITTF project. The combined system will have the potential to perform or augment sensing, spraying, pruning, thinning, and harvesting in orchards, although full scale automation of all above tasks in a commercially viable fashion may be far in the future.

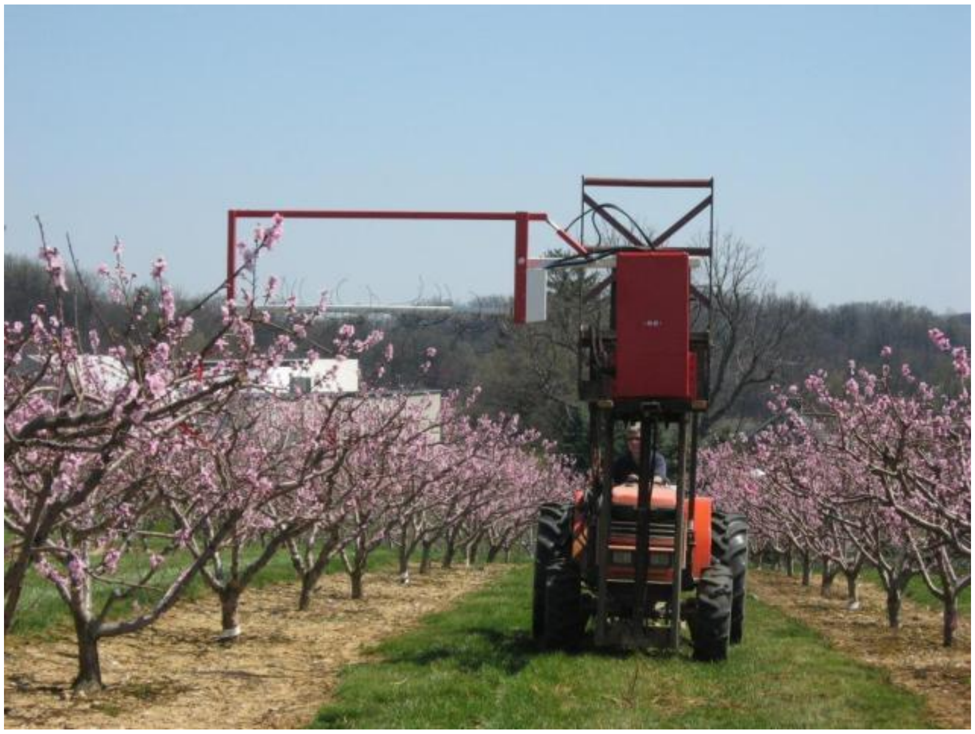


Figure 2.1 Darwin string thinner attached to a tractor during operation on vase-shape peach trees (Courtesy of Tara Baugher from PSU).

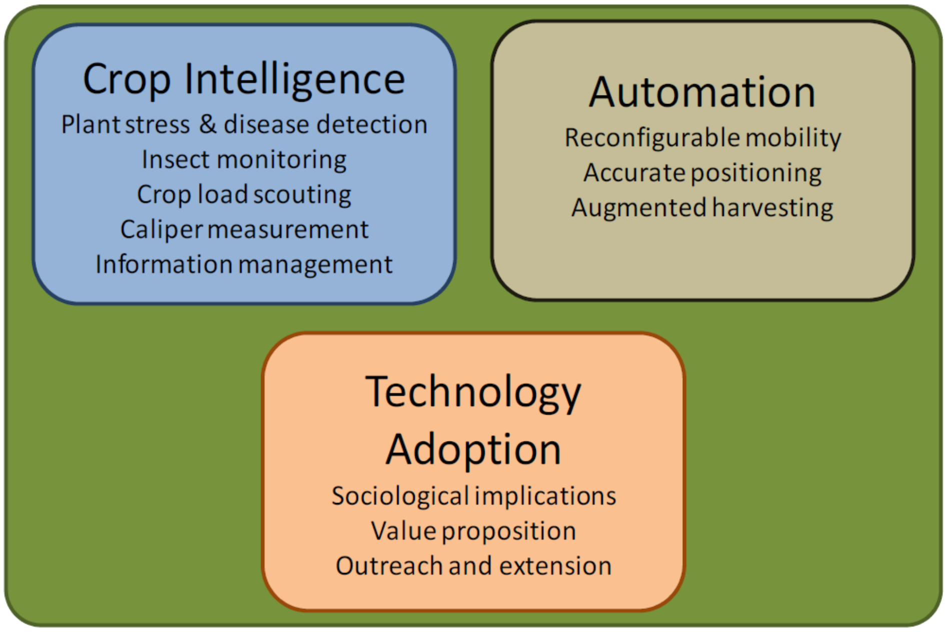


Figure 2.2 Three main themes and underlying topics of CASC project (Hamner et al., 2011).

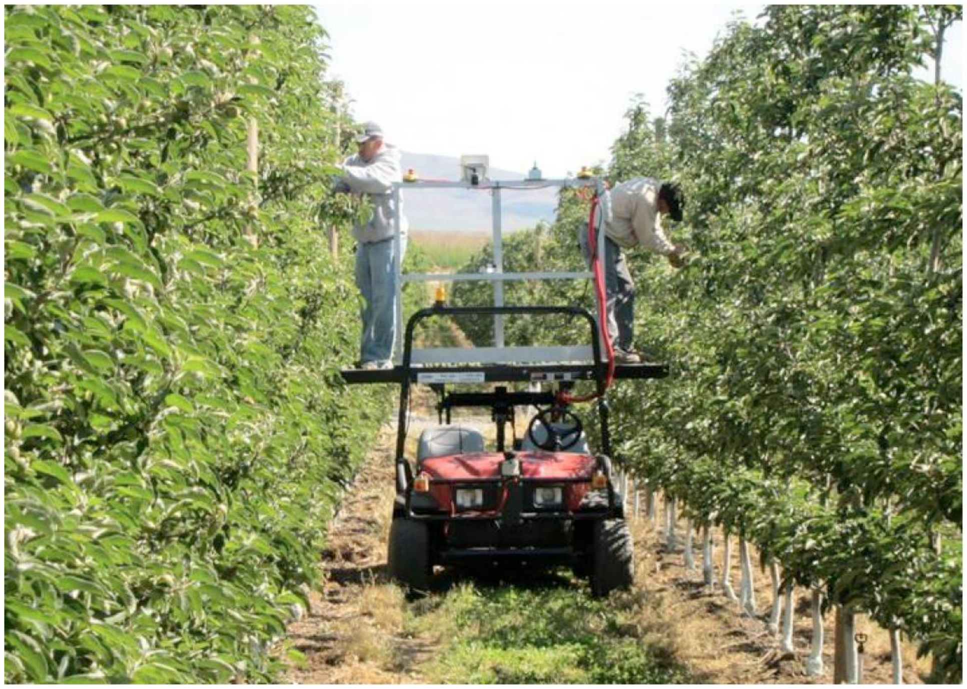


Figure 2.3 Farmers can perform thinning, pruning, or other operations on the platform of an autonomous vehicle developed by CASC project (Bergerman et al., 2012).

2.2 Definition of a Robot

The word “robot” originates from the Czech word “robota,” which means work. Since Karel Čapek introduced the term robot in his play “Rossum’s Universal Robots” in 1920, the term has been widely applied to a variety of mechanical devices (Spong et al., 2006). Almost any device that operates with some degree of autonomy can be regarded as a robot, from autonomous cars, Unmanned Aerial Vehicles (UAVs), the self-guided vacuum cleaner Roomba®, Sony®’s robotic dog AIBO, Honda®’s humanoid ASIMO, to the Mars Exploration Rover developed by NASA.

Among the range of devices that can be termed “robots,” the first one realized in modern society was a mechanical arm with a gripper, named Unimate. It was invented by American inventor George Devol and installed in a General Motor’s plant for die casting in 1961 (Corke, 2011). This arm-type robot and its descendants formed a subclass termed “manipulators” in robotics.

A manipulating industrial robot is defined in the ISO 8373:1994 standard as “an automatically controlled, reprogrammable, multipurpose manipulator, programmable in three or more axes, which may be either fixed in place or mobile for use in industrial automation applications” (ISO, 1994). In this thesis, the term robot refers to an automated device comprised of a computer-controlled manipulator of the type shown in Figure 2.4 unless otherwise specified.

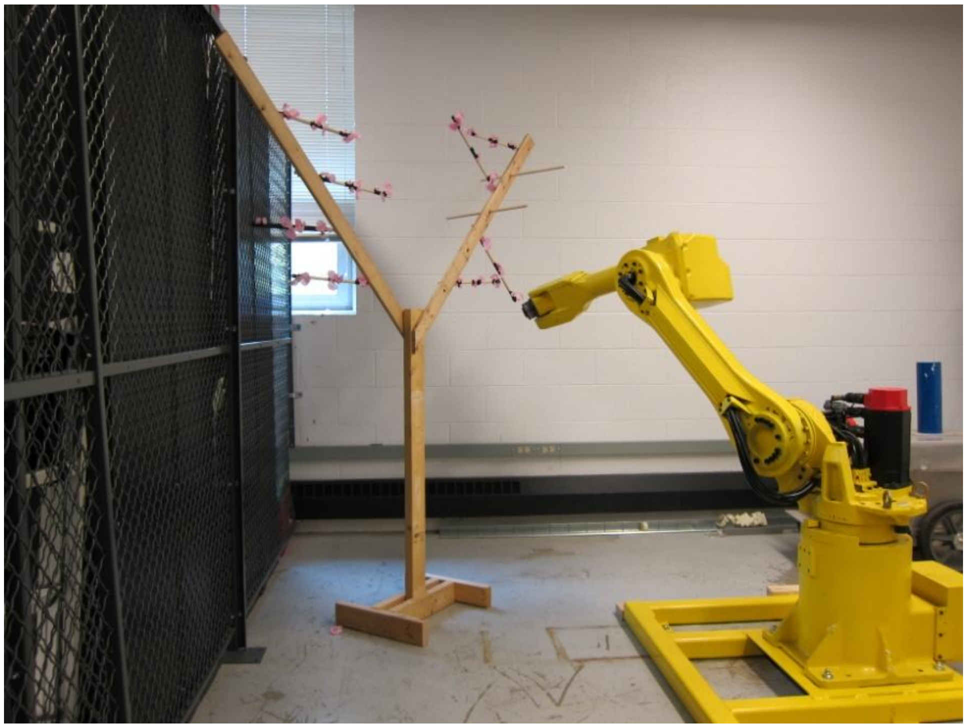


Figure 2.4 FANUC M-16iL industrial-grade manipulator with a model tree.

2.3 Robotics in Agriculture

Since the debut of manipulators in 1961, assembly lines in which large numbers of highly repetitive and labor-intensive tasks are required have been the major operating realm for industrial robots. However, the less structured and uncertain nature of agriculture was a hurdle for robot deployment. The other resistance was the lack of interest and investment. By searching keywords “agricultural robot”, “medical robot”, “aerial robot”, “space robot”, and “service robot” on the searching engine Google Scholar for research publications during the period from 2000 to 2011, the number of hits (Figure 2.5) shows that robotics in agriculture represents a minority compared to its counterparts in other fields. In section 2.3.1, the proliferation of robotics in agriculture is briefly discussed.

A manipulator is merely one type of existing agricultural robot or, in most cases, just one component working with other units such as moving platforms in a robotic system. Since this arm-type device in plant production is of special interest in this study, some applications of manipulators are reviewed in section 2.3.2.

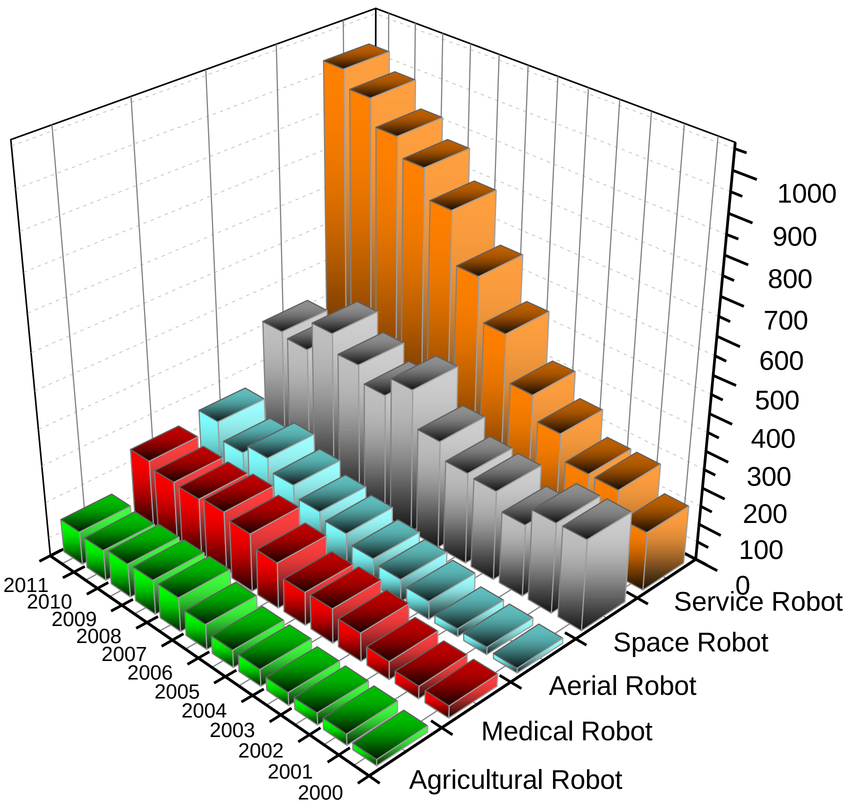


Figure 2.5 Keyword hits of robots in different applications since year 2000.

2.3.1 Proliferation of Robotics in Agriculture

Mechanization and automation is a trend in agriculture that started in the 1940s. When intelligence was added to agricultural machinery, the era of agricultural robotics started. There has been ample effort in proliferating robotics into various aspects of agriculture in the past three decades. An overview of robotics in agriculture was provided by Grift et al. (2008). Most of the applications were in plant production, such as grafting, transplanting, tilling, seeding, cultivation, fertilizing, spraying, harvesting, and related information management for enhancing the performance of above tasks. Kondo et al. (2011) summarized the research to date, and introduced technology developments from the viewpoints of mechanism, vision, end-effectors and arm, as well as vehicle automation. A grafting robot for cucumber was a successful example for commercialization. Starting from specific varieties of cucumbers in Japan, it spread to other countries in East Asia, Europe, and now North America (Kubota et al., 2008) for different plants.

The most successful, and in fact the only majorly commercialized robot in agriculture to date, is the milking robot. This technology allowed dairy cows to be milked *ad libitum* without human intervention. Since the first installation of automatic milking system in the Netherlands in 1992, about 2200 farms worldwide adopted this technology by the end of 2003 (De Koning and Rodenburg, 2004). In December 2011, one of the milking robot manufacturers, Lely, celebrated the 12,500th installment of their product (Lely, 2011). As Kondo et al. (2011) indicated, ample research was devoted to vehicle automation. The idea was to replace drivers from common agricultural vehicles such as tractors, transplanters and “gators” through autonomous steering units. Among the efforts, some were focused on smaller moving platforms to serve as scouts for environmental factors, or as carriers for manipulators.

2.3.2 Manipulators in Harvesting

The most common application of manipulators in agriculture is harvesting among plants, such as grapes (Monta et al., 1995), tomatoes (Kondo et al., 1996), melons (Edan et al., 2000), cucumbers (Van Henten et al., 2002), and apples (Baeten et al., 2007). Robotic thinning has never been a main objective in plant production, although it is similar to fruit harvesting: Firstly, both tasks require a computer vision unit to identify the location of target objects, which were fruits and blossoms. Occlusion caused by leaves, stems, or branches, as well as occlusion by the targets themselves is also a common theme in the associated computer vision task. Secondly, both tasks require manipulators to move end-effectors toward target objects in an anthropomorphic manner. In addition, the computer vision unit and the manipulator need a carrier to transport them in the facilities or fields. The carriers could have various forms, such as wheel type (Kondo et al., 1996), crawler type (Monta et al., 1995), rail type (Van Henten et al., 2002), or a trailer attached to a tractor (Baeten et al., 2007).

Monta et al. (1995) applied a five-DOF (degree-of-freedom) manipulator comprised of a spherical arm with two rotational joints as a wrist for operations in a vineyard. Four dedicated end-effectors were developed for harvesting, berry thinning, spraying, and bagging, and they demonstrated the versatility of a single manipulator in an agricultural production system. The motivation of this study was not only to alleviate the requirement of skilled personnel but also to improve exhausting work conditions. In Japan, most of the grapes were cultivated in trellis systems, where canopies are supported by metal structures at a height of 170-190 cm. In this environment, farmers have to keep their hands overhead for hours when working, which is unacceptable from an ergonomic point of view. This case could be taken as a supporting example for the idea of developing a fully automated orchard system.

Van Henten et al. (2002) applied a Mitsubishi RV-E2 robot, a six-DOF manipulator with all revolute joints for cucumber picking. By mounting the manipulator on a linear mobile platform (Figure 2.6), the robot reached seven DOF which increased its operational space. To accommodate the robot in the work environment, an innovative “high-wire” cultivation system was introduced. The overall accuracy of cucumber-picking was 80% at a picking rate of one vegetable harvested in 45 seconds. The insight form this application was that the horticultural and engineering techniques should evolve together when pursuing a breakthrough in production.

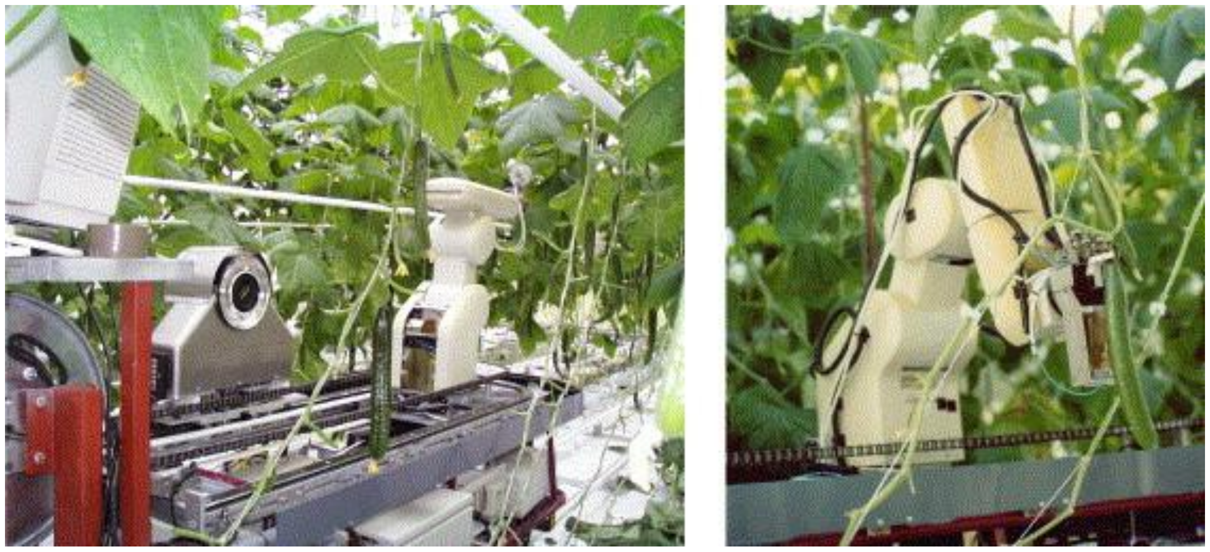


Figure 2.6 Field test of a cucumber harvesting robot (Van Henten et al., 2003b).

Baeten et al. (2007) harvested apples in an orchard using a Panasonic VR006L industrial manipulator, mounted on a tractor (Figure 2.7). To avoid the negative influence of changing ambient light, a tent structure was used to cover the robotic system and a row of fruit trees. Although the overall speed or the time allotment of each cycle was not reported, the system reached an 80% picking rate in specific experiment setups. However, the size of the robotic system, the limited height of apple trees, and the use of tent structure were not satisfactory and should be improved significantly for real deployment.

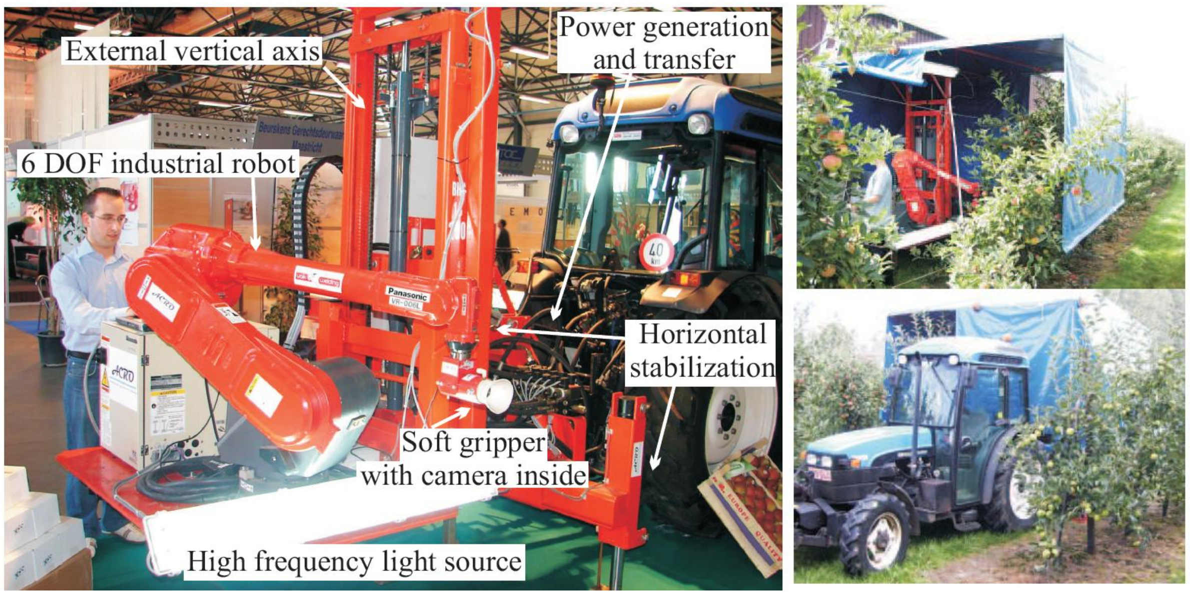

External vertical axis
Power generation and transfer
6 DOF industrial robot
Horizontal stabilization
Soft gripper with camera inside
High frequency light source

Figure 2.7 An industrial manipulator driven by a tractor (Baeten et al., 2007).

The cases discussed above represent indoor operations, which excludes thinning. In outdoor operations, the intensity and frequency spectrum of incident sunlight varies significantly over the course of a day. Insufficient ambient light may be alleviated by adding artificial light sources mounted on a carrier or a manipulator. However, excessive lighting causes overexposure, and requires extra calibration procedures for correction, a procedure that slows down the robotic thinning system and deteriorates its overall performance. In addition, the incident angle of sunlight varies as well, causing inconsistent shadows, which complicates performing compensation by using *a priori* knowledge.

Wind can pose further negative effects in outdoor scenarios, in that it oscillates the blossoms and branches, changing the positions of target objects. In this scenario, the planned path to a target blossom might be invalid or not collision-free. These changing environmental factors require a robot to reach a balance between performance and robustness.

The actions of the end-effectors in thinning are less complicated than during harvesting, because in thinning merely a removal action is required while during harvesting sometimes a complicated mechanism of wrist movements is required. For instance, in tomato harvesting, the fruit (although deemed a vegetable by Supreme Court ruling) is twisted off the vine, mimicking a human harvester. However, in most cases of fruit harvesting, the orientation is well defined, since most fruits are hanging vertically. This is not the case in thinning, since the blossom is arbitrarily oriented along the circumference of a branch. One method to approach this problem is by requiring the machine vision system to accurately measure the orientation of the branch, along with the three dimensional coordinates of the blossom, which is a daunting task. An implicit assumption in this research was that the end-effector is robust with respect to the orientation of the blossom along the circumference of the branch. Section 3.4 elaborates on this principle.

2.4 Task Planning Algorithm for Manipulators

At the start of the robotics era, these machines were merely used for highly repetitive tasks such as placing car doors on an assembly line, welding seams and painting. These types of operations consist of a fixed sequence of commands such as Move, Grasp, Pickup, and Putdown, and the robots usually did not perceive the environment. The next level of advancement originated when the robots were given some cognitive intelligence enabling them to respond to varying situations. This ability requires a sensor-based control system equipped with a task planning algorithm. Tung and Kak (1996) used two cameras to simultaneously acquire coarse and detailed information of randomly oriented parts, and assembled them using a PUMA 762 six-axis manipulator.

Edan et al. (1991) developed a near-minimum-path algorithm for visiting N given locations on a citrus tree. The sequence of visiting points was determined by solving the traveling salesman problem with kinematics and inertial parameters of a cylindrical manipulator. By dividing a tree into several subvolumes and estimating distances among points, the algorithm could decide a visiting sequence for 250 fruits in 1.013 seconds. However, the path was not guaranteed to be collision-free and the orientation of the end-effector was not considered. In addition, some fruit trees such as peach can be trained into a relatively simplified structure (section 3.1.1), and hence the visiting sequence can be as simple as a top-down branch-wise routine. This is an example of how a horticultural operation can reduce the complexity of engineering problems.

Scant literature was discovered regarding manipulators in plant production addressing the issue of task planning. Van Henten et al. (2003a) provided a methodology for generating a collision-free motion of a cucumber picking robot as shown in Figure 2.8. The methodology was used as a framework for the proposed task planning algorithm in this study.

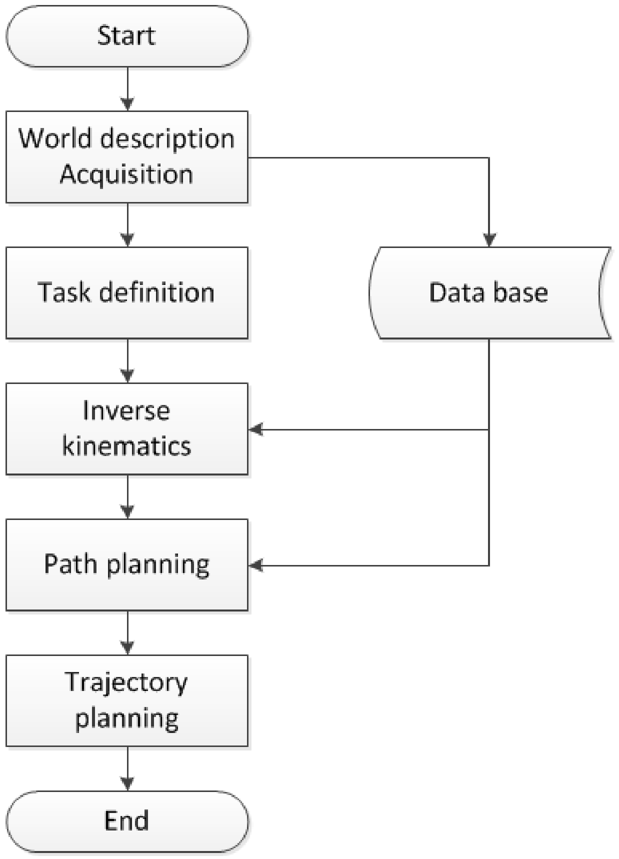


Figure 2.8 Methodology of generating a collision-free path. A computer vision unit provides the information of environment and target objects. Combined with *a priori* knowledge, the task planning system provides the final trajectory as output. Adapted from Van Henten et al. (2003a).

To eliminate potential ambiguity of terminology, it is helpful to clarify some terms used in this thesis. “Path planning” refers to an algorithm that provides a collision-free path from any obstacles for a manipulator using a set of initial conditions and goal configuration as inputs. A path is a locus described either in the configuration space (joint space) or the operational space (Cartesian coordinate system), and it only explains a manipulator’s geometrical motion. “Trajectory planning” takes velocity, acceleration, and smoothness constraints into account, and refines the path provided by the path planner. Sciavicco and Siciliano (2000) disambiguate these two terms in a similar way. “Task definition” in this thesis refers to a process which converts input arguments into standard format for subsequent calculations. This conversion is not a one-to-one transformation and involves decision making procedures. Task planning, which is highlighted in Figure 1.2, is a higher level concept. It includes the task definition, inverse kinematics calculation, path planner, and trajectory planner as shown in Figure 2.9. In this thesis, task definition is of special interest, while the path planner and trajectory planner are left to future research.

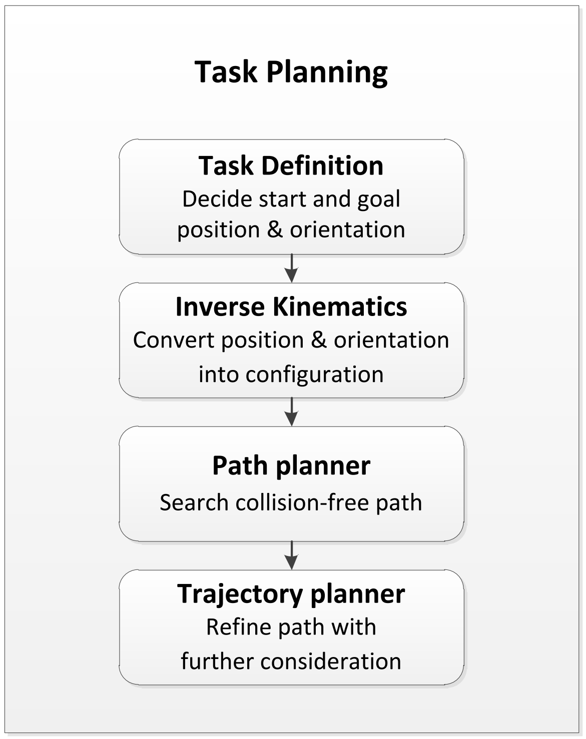

Task Planning
Task Definition
Decide start and goal position & orientation
Inverse Kinematics
Convert position & orientation into configuration
Path planner
Search collision-free path
Trajectory planner
Refine path with further consideration

Figure 2.9 Concept of tasking planning and its components.

3 MATERIALS AND METHODS

This chapter elaborates on how a fruit tree was represented by an artificial model tree that was used for verification experiments as discussed in section 3.1. Section 3.2 is devoted to the introduction of hardware and software for a custom robot developed for this study. Section 3.3 describes how the end-effector functions, and how its representative version works. Section 3.4 and 3.5 cover the four components of task planning illustrated in Figure 2.9.

3.1 Tree & Blossom

To build a model tree, there are two assumptions that must be clarified. (1.) What cultivar of tree does the model tree represent? (2.) What is the distribution or clustering pattern of the blossoms on the tree? These matters are discussed in subsection 3.1.1 and 3.1.2. The final design of the model tree can be found in subsection 3.1.3.

3.1.1 Selected Peach Cultivar

There are three important terms that are commonly confused being species, variety, and cultivar. All cultivars of peach are the same “species”: *Prunus persica* (L.) Batsch. “Varieties” occur through natural selection, while “cultivars” refer to varieties selected by human intervention. In commercial peach production, a variety of cultivar is found such as Saturn, White Lady, PF-17, Fantasia, and NJF16, each with unique blossom cluster patterns. The Saturn variety cultivar (Figure 3.1) was selected for this study since ample experience has been gained through research at PSU.

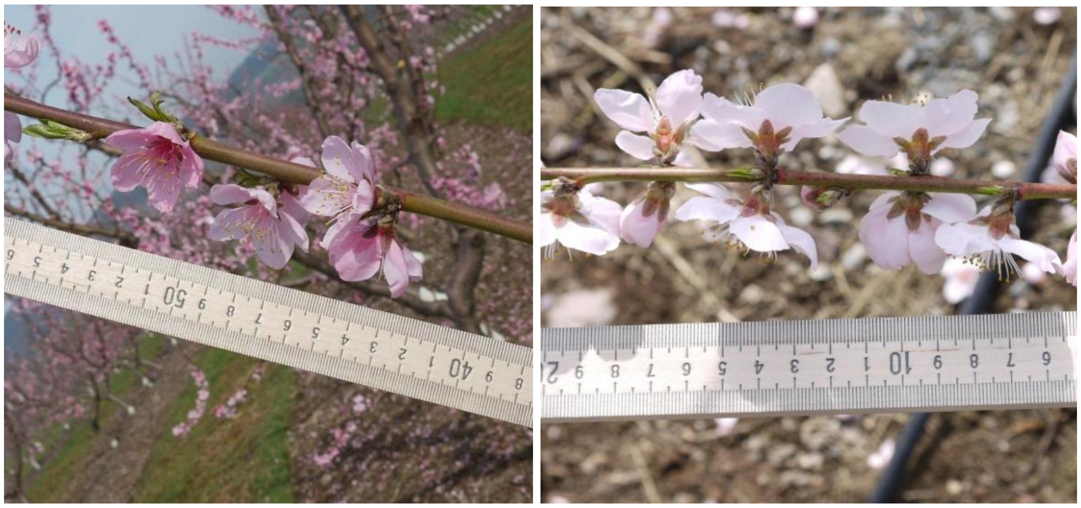


Figure 3.1 Left: A branch with blossoms from Saturn. Right: A branch with blossoms from NJF16. Different cultivars have varying colors and cluster patterns of blossoms (Courtesy of Dr. James Schupp from PSU).

Peaches are generally grown for about 2 years before cropping and they reach a full size in five seasons. Before a tree contributes to the production, many steps of training should be performed. It is common to train the trees into a preferred shape by pruning and other treatment. The KAC-V system (DeJong et al., 1994) is widely adopted and was the most common form applied in research activities related to this study. Trees of this type are trained into a V-shape by leaving two scaffolds as shown in Figure 3.2. Ideally, each scaffold should have 25° to 40° deviation from a vertical axis. Another alias of this type was “perpendicular-V,” which implies that the V-shape is perpendicular to the moving direction of a machine in a row, rather than having a right angle between scaffolds. In this study, only mature trees were considered because they constituted the largest portion of production.

A mature peach tree has a height of approximately four meters. After training, only two scaffolds are left, where each scaffold has approximately 30 fruiting branches. The branch orientation started from horizontal (90°) to nearly vertical (10°). If a certain branch has a less accessible orientation, pruning is applied to correct it. An idea inspired by this fact was that specific pruning treatment can be developed to facilitate the automated selective thinning system. This agreed with the research conducted by Van Henten et al. (2002), who introduced a new cultivating convention to deploy a robotic harvesting system.

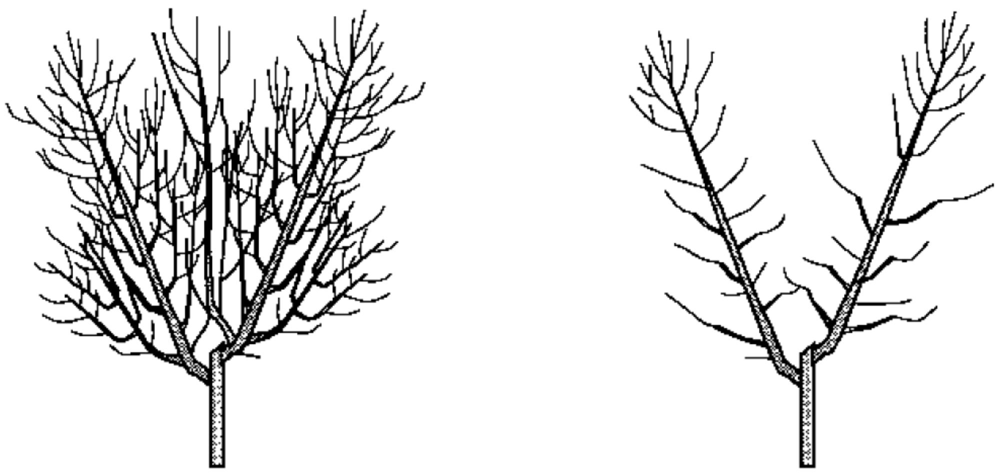


Figure 3.2 Left: Peach tree without training. Right: Peach tree trained into a KAC-V shape (DeJong et al., 1994).

3.1.2 Blossom Thinning Pattern

Instead of providing a general description regarding how blossoms are distributed on Saturn, an alternative way was to provide a blossom thinning pattern. A rule-of-thumb for thinning Saturn is called the “5-8 pattern” as illustrated in Figure 3.3. Here all blossoms along the first 5 cm from the origin of the branch are removed. Then, from the 5-cm point, it divides the branch into alternating sections of 5 cm and 8 cm respectively until the branch end is reached. After removing the blossoms in the 0~5 cm window, in every subsequent 5 cm window, only two blossoms are kept: the remaining ones are removed with the ones located inside any 8 cm window. The blossoms left in the 5 cm windows should be separated as far as possible, so usually the ones located in close proximity to the extremes of the interval are kept.

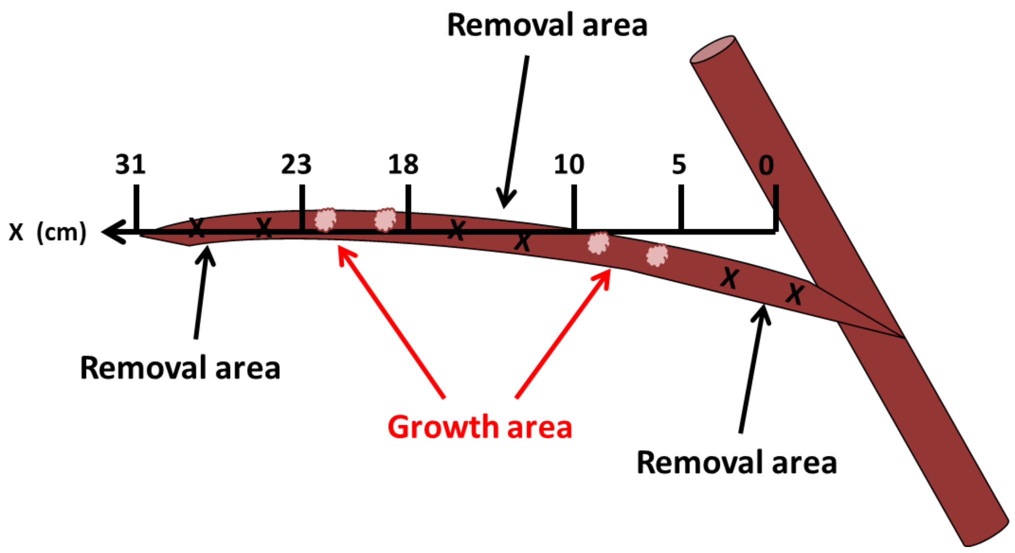
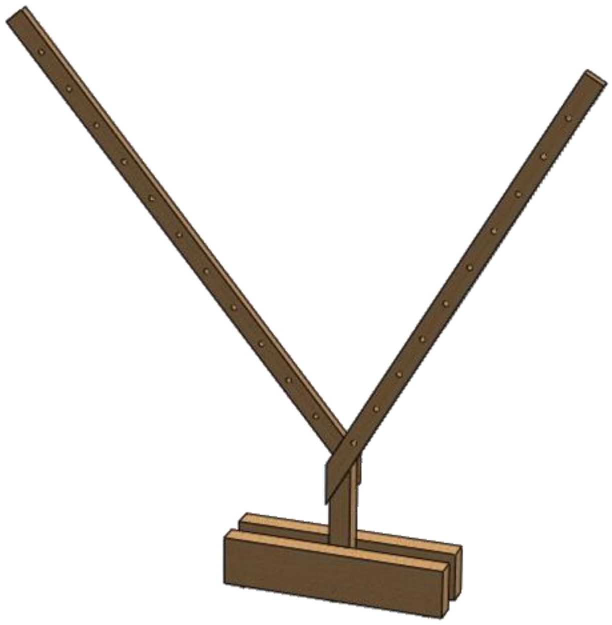


Figure 3.3 Schematic of the “5-8 pattern.” Starting from the 5-cm point, the branch is separated into alternating windows of length 5 cm and 8 cm respectively, until the branch end is reached (Courtesy of David Lyons from PSU).

3.1.3 Model Tree

In general, a mature tree has a height of about four meters. To build a counterpart for the scaled-down manipulator, a model tree was designed with a height of 100 cm as shown in Figure 3.4. The trunk was made from standard lumber while the two scaffolds were made from thinner slats. The right scaffold deviated from the vertical axis by 30°, and the left by 40°. The design reflected the fact that real trees are usually asymmetrical, and also provided a more representative model for future experimental needs. A detailed drawing and dimensions can be found in Appendix A. Since the manipulator would not thin two scaffolds simultaneously, only the right scaffold was attached to save space.



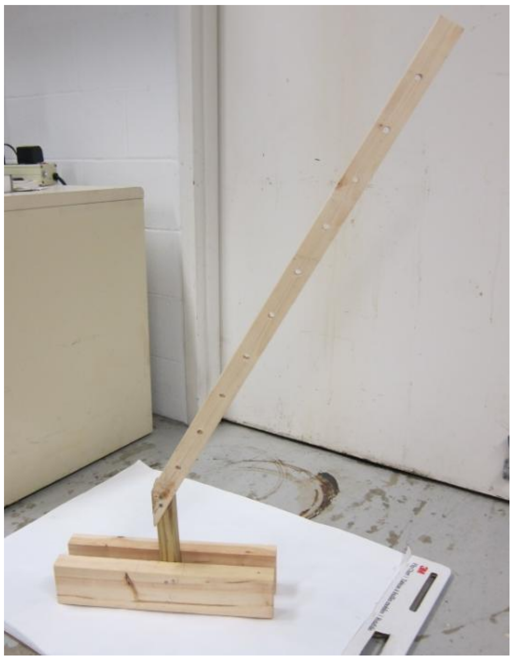


Figure 3.4 Left: model tree designed using CAD software SolidWorks®. Right: prototype of the design.

The above model is just a frame of a peach tree since the branches are not included. Since the branches have diverse orientations, curves, and lengths, they are difficult to represent by standard lumber using dowels and strips. As an alternative, Loc-Line® ¼'' modular hose was used (Figure 3.5); it is manufactured by Lockwood Products, Inc. from Lake Oswego, Oregon. This hose is composed of separable units and can be extended to arbitrary lengths. In addition, the joint design of each unit supports the hose in a variety of poses, which can be used to represent branches. After assembling, each unit had a length of about 1.5 cm. The smallest diameter outside the cone-shape unit is about 1 cm while the largest is about 1.6 cm. The blossoms were represented by round stickers attached to the hose, yielding adjustable locations according to experimental needs. More information about the stickers can be found in section 4.2.1.

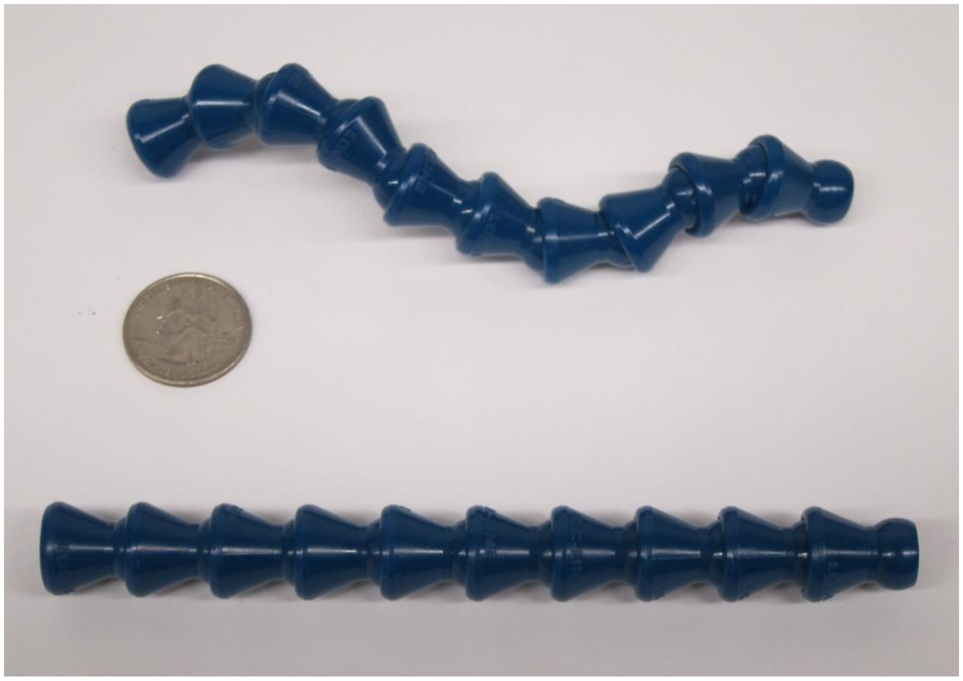


Figure 3.5 Loc-Line® ¼’’modular hose with a quarter for size comparison.

Although the variety of orientations can be represented by this design, the flexibility of real branches cannot. Flexibility of branches causes target blossoms to vibrate when perturbed by external influences such as wind and inadvertent contact by the end-effector. This mismatch is partially addressed by the design of a clamp-type end-effector, which has a high tolerance for inaccurate positioning. Therefore, vibrations were not taken into account in this study.

3.2 Manipulator

At the start of the project, an industrial-grade manipulator FANUC M-16iL as shown in Figure 2.4 was procured from a dealer of refurbished robots. However, this manipulator was originally built for repetitive tasks, and the real-time communication with a computer and potentially external sensors was limited. The decision was made to reduce the problem to a more manageable size by developing a scaled-down version of the robot and to set the target of proving the concept as the research goal. In the new scenario, a tabletop robot was required. Most tabletop models have a limited payload and for this reason, a custom manipulator (Figure 3.6) capable of lifting and moving a rather heavy end-effector was built in a joint effort among researchers from PSU and UIUC. The design was completed using SolidWorks® (Dassault Systèmes SolidWorks Corp., Vélizy, France).

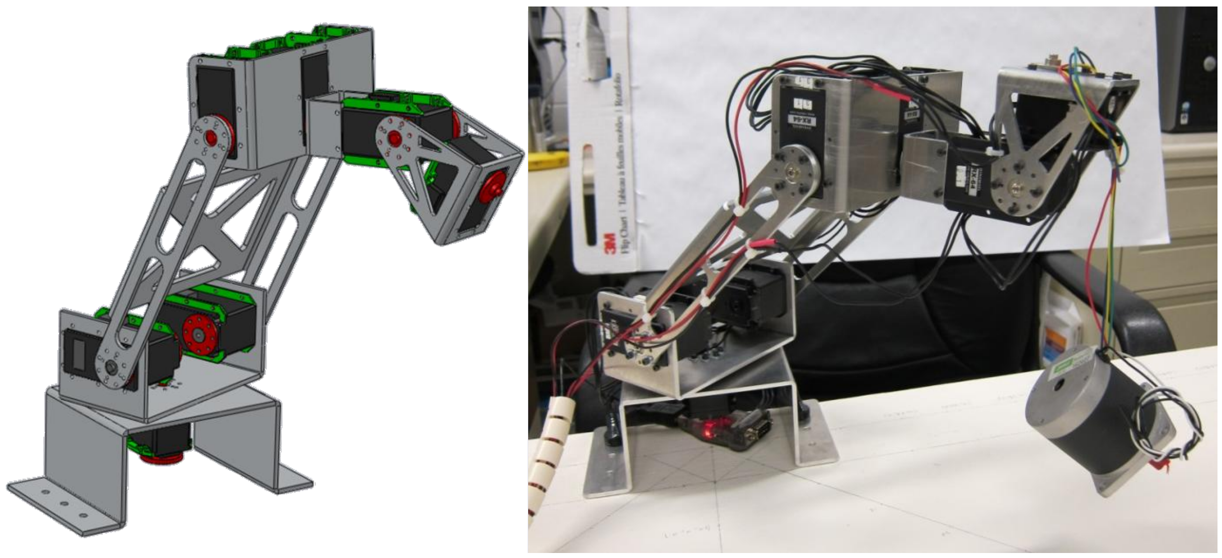


Figure 3.6 Left: CAD drawing of the custom manipulator in SolidWorks®. Right: custom manipulator lifting a 650 g stepper motor as a significant weight.

3.2.1 Hardware

The manipulator joints were revolute and actuated by servomotors (ROBOTIS, Seoul, Korea). As shown in Figure 3.7, these motors were connected using a RS-485 network bus allowing a single controller to drive all motors. Three models being RX-28, RX-64, and EX-106+, were chosen to meet the various torque requirements of each joint. All the parts of the manipulator's body were made from aluminum alloy 6011 (AA6011), and machined by the Physics Machine Shop at PSU and the Mechanical Engineering Machine Shop at UIUC.

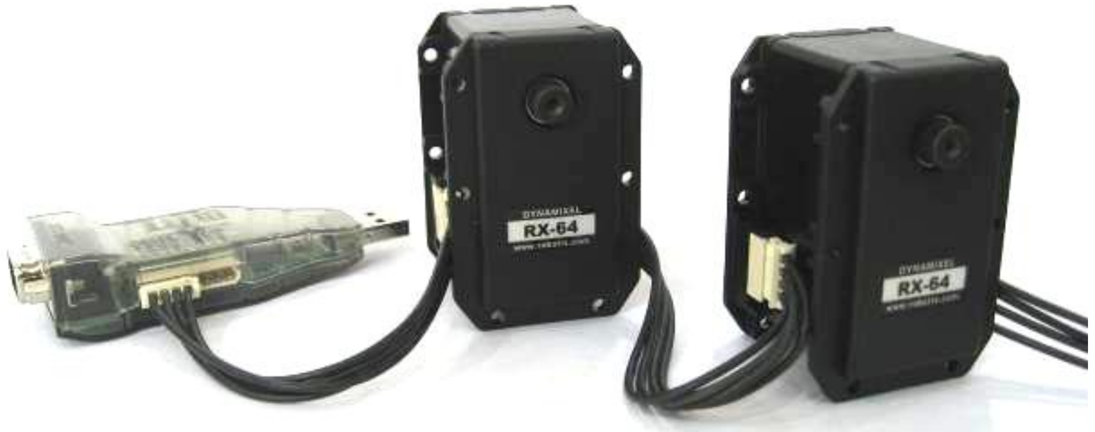


Figure 3.7 ROBOTIS® motor controller USB2Dynamixel and two networked servomotors RX-64 (ROBOTIS, 2006).

As shown in Figure 3.8, this robot is a six DOF all-revolute-joint manipulator, categorized as an articulated manipulator with a spherical wrist. The latter was an important concept especially for computing inverse kinematics, which is discussed in section 3.5. The Denavit-Hartenberg (DH) convention was introduced to describe the reference frames and configurations.

Note the existence of two DH conventions, an original and a modification, as discussed by Corke (2011). The original notation was adopted here, and the reader could verify it by checking the subscripts of the DH parameters provided in Table 3.1. A fixed offset of 90° was assigned to joint angle θ_3 for setting the full extended configuration as the "zero position." This configuration has only zeros as revolving angles to all joints. The 90° offset was coded in the control software, so users did not need to concern themselves with this value when moving the manipulator.

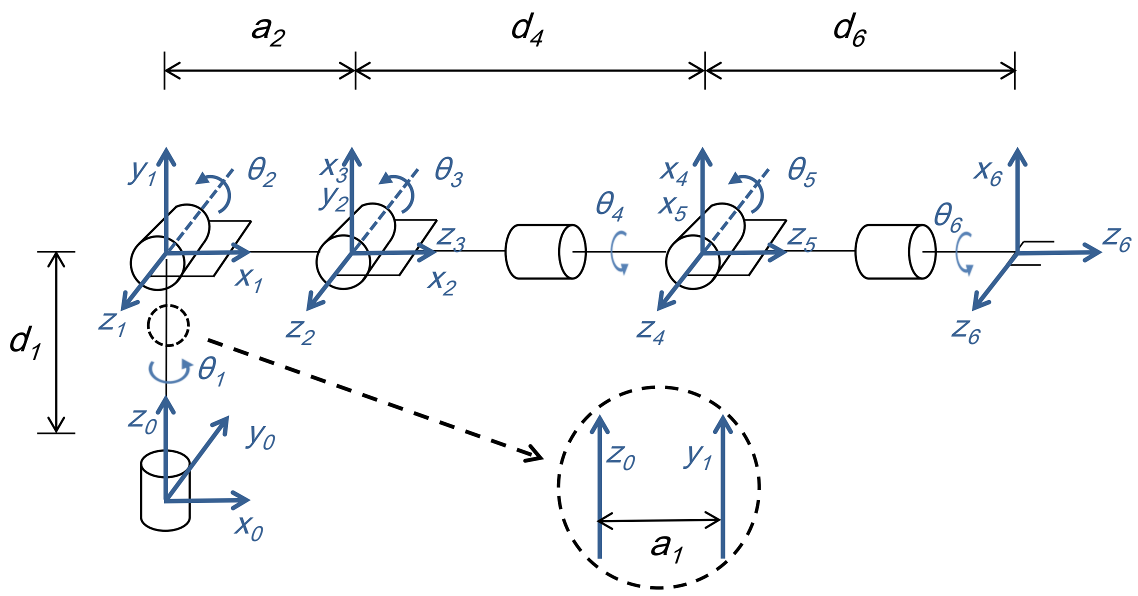


Figure 3.8 Schematic of the custom manipulator with frame assignment. This is the zero position according to the DH parameters displayed in Table 3.1. Note that the axis of z_0 and y_1 are not coincident due to the existence of a small link length a_1.

Table 3.1 DH parameters for the manipulator.

Link #	Link length a (cm)	Link twist α (deg)	Link Offset d (cm)	Joint angle θ (deg)
1	a_1	+90°	d_1	θ_1^*
2	a_2	0	0	θ_2^*
3	0	+90°	0	$\theta_3^* + 90°^\dagger$
4	0	-90°	d_4	θ_4^*
5	0	+90°	0	θ_5^*
6	0	0	d_6	θ_6^*

* Joint variable

† Fixed offset for the zero position

The parameters with symbols *a* and *d* in the DH table are constants and inherent to the manipulator's design, while the θ 's are joint variables varying with configurations. Each row in Table 3.1 represents a homogeneous transformation A_i, which is composed of four successive basic transformations (eq. 1) being rotating about the current z-axis for θ ($Rot_{z,\theta}$), translating along the current z-axis for *d* ($Trans_{z,d}$), translating along the current x-axis for *a* ($Trans_{x,a}$), and finally rotating about the current x-axis for α ($Rot_{x,\alpha}$). The matrix A_i transforms any point *p* from its current coordinate frame to a new one by calculating $p' = A_i \cdot p$. By multiplying A_i one after another, the final transformation matrix T_6^0. which transforms the coordinates from the world frame (frame 1st) to the tool frame (frame 6th) can be solved. Again, note the definition of the equation. The original and modified DH conventions have a different order and elements for matrix multiplication. Spong et al. (2006) elaborated on the structure and use of the transformation matrix. In this thesis, sin θ and cos θ *are* expressed as c_θ *and* s_θ to save space.

$$A_i(\theta_i) = Rot_{z,\theta_i} Trans_{z,d_i} Trans_{x,a_i} Rot_{x,\alpha_i} \qquad (1)$$

$$A_i(\theta_i) = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_i(\theta_i) = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} c_{\alpha_i} & s_{\theta_i} s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i} c_{\alpha_i} & -c_{\theta_i} s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \underset{3\times3}{R_i} & \underset{3\times1}{t_i} \\ 0\ 0\ 0 & 1 \end{bmatrix}$$

In Table 3.2, the values of non-zero DH parameters are listed, except the joint variable θ's. Although the theoretical values can be acquired from the design drawings of the manipulator, the real values can be slightly different owing to the machining errors. In this design a_3 and d_4 are interchangeable; in other words, either a_3 or d_4 has a non-zero value.

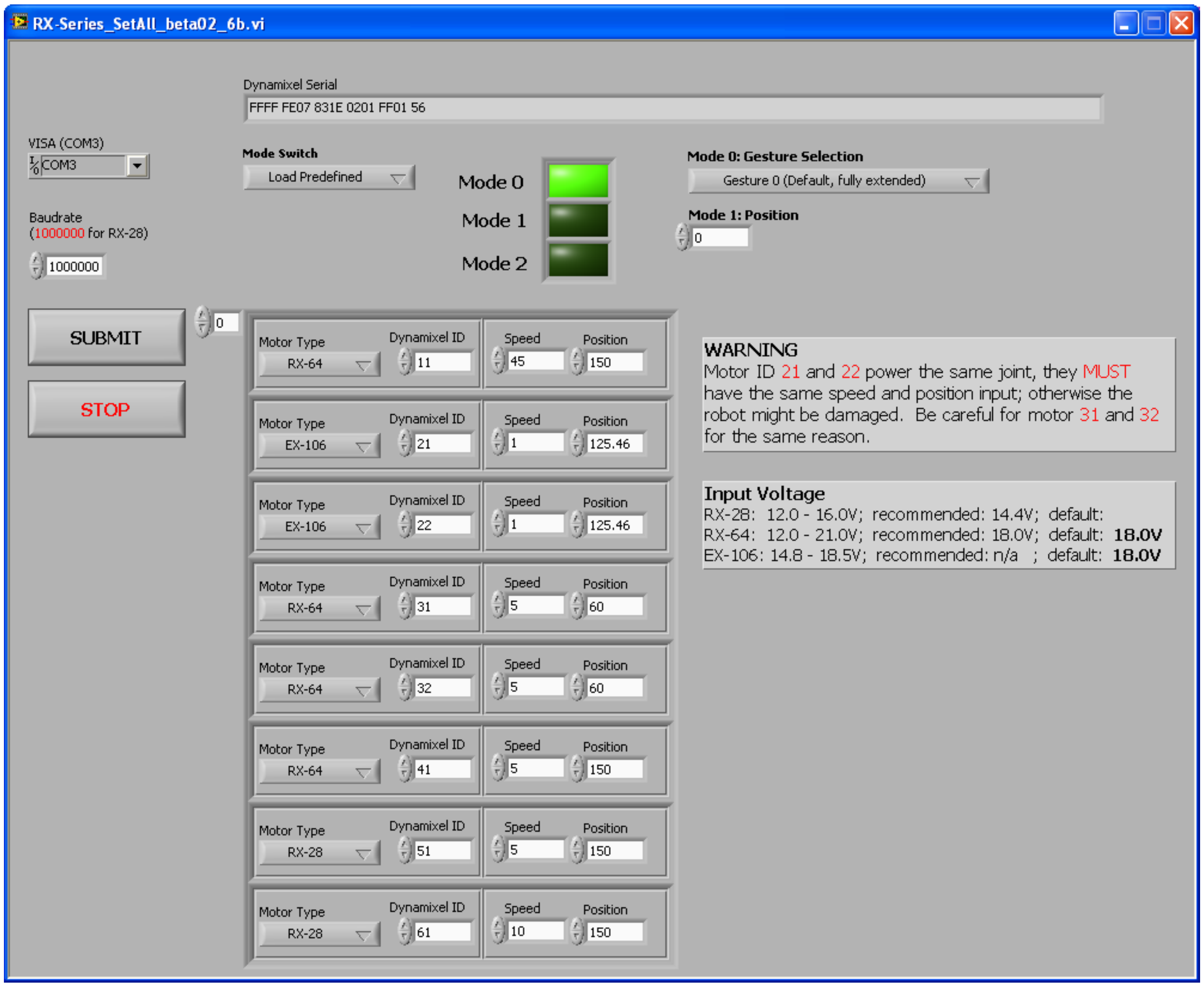
Table 3.2 Calibrated values of non-zero DH parameters.

Variable	a_1	a_2	d_1	d_4	d_6
Length (cm)	1.35	17.75	11.4	10.25	6.4

3.2.2 Software

The control software of the manipulator was written in MATLAB® (MathWorks, Natick, Massachusetts) version 7.11 (R2010b) using the Symbolic Math Toolbox™ version 5.5 (R2010b). The actuation of the servomotors required manufacturer’s driver files, which were written in C and executed as a MATLAB subroutine. Further explanation and implementation code can be found in Appendix B.

For ease of calibration and testing, a control program (Figure 3.9) was written in LabVIEW® (National Instruments Co., Austin, Texas) version 2011 (version 10.0) at the development stage. Although it was later replaced by the more powerful MATLAB® version as mentioned above, it is occasionally used for joint-by-joint calibration due to its friendly GUI and straightforward dataflow display. When running this tool, shutting down MATLAB is recommended because they compete for I/O port access. Dyamixel Manager version 0.98 from ROBOTIS® is helpful when inspecting individual servomotors, such as instantaneous position, speed, torque, temperature, voltage, and current, or when performing motor-by-motor calibration. Note that calibration on a joint-by-joint level is different from the motor-by-motor level since joint 2 and 3 are powered by dual motors.



RX-Series_SetAll_beta02_6b.vi

Dynamixel Serial: FFFF FE07 831E 0201 FF01 56

VISA (COM3): COM3

Baudrate (1000000 for RX-28): 1000000

Mode Switch: Load Predefined

Mode 0
Mode 1
Mode 2

Mode 0: Gesture Selection: Gesture 0 (Default, fully extended)

Mode 1: Position: 0

SUBMIT

STOP

0

Motor Type	Dynamixel ID	Speed	Position
RX-64	11	45	150
EX-106	21	1	125.46
EX-106	22	1	125.46
RX-64	31	5	60
RX-64	32	5	60
RX-64	41	5	150
RX-28	51	5	150
RX-28	61	10	150

WARNING
Motor ID 21 and 22 power the same joint, they MUST have the same speed and position input; otherwise the robot might be damaged. Be careful for motor 31 and 32 for the same reason.

Input Voltage
RX-28: 12.0 - 16.0V; recommended: 14.4V; default:
RX-64: 12.0 - 21.0V; recommended: 18.0V; default: **18.0V**
EX-106: 14.8 - 18.5V; recommended: n/a ; default: **18.0V**

Figure 3.9 Calibration tool for joint-by-joint inspection written in LabVIEW®.

3.3 End-effector

The task of the end-effector is to remove target blossoms from branches. To accomplish this goal, an investigation of various designs was conducted by PSU and a full-scale prototype was developed. The control program was developed by UIUC. However, this end-effector was not a scaled-down version, and hence a representative model, which had a scaled-down size, was built by UIUC. This representative model was connected to the custom manipulator for verification of the task planning algorithm. The full-scale prototype is briefly introduced in section 3.3.1, and the representative model is discussed thereafter.

3.3.1 Full-scale End-effector

Among a variety of potential blossom removing end-effectors based on air blasts, laser beams, water jets, and mechanical methods, the device chosen for optimal robustness and simplicity consists of a clamp-type device containing dual rollers that brush off the blossoms. Figure 3.10 shows the device, which is powered by two RX-28 servomotors and can be directly linked to the control network of the manipulator. The performance of the end-effector will be evaluated and reported by PSU in the future.

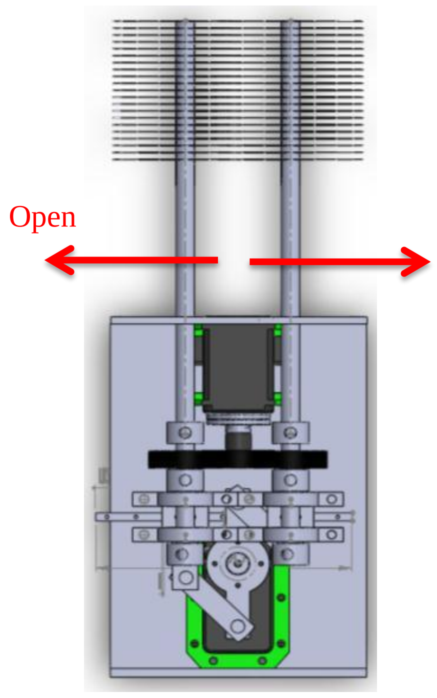


Figure 3.10 Design of the end-effector (Courtesy of David Lyons, PSU).

The actions of the end-effector are shown in Figure 3.11. Given the coordinates of the blossom and its orientation of the branch, the end-effector enters a standby posture, and adjusts the orientation itself at the first step. In step 1, the two forks of the end-effector move along the branch and stop at the thinning position. Although not shown in the representative model, the brushes attached to the two forks remove the blossoms and they spin in step 2. Before the spinning begins, a sliding mechanism closes the forks; therefore, the end-effector resembles a clamp holding the target. In the third step, the forks open to the original position, and the end-effector retreats from the thinning posture to the standby posture.

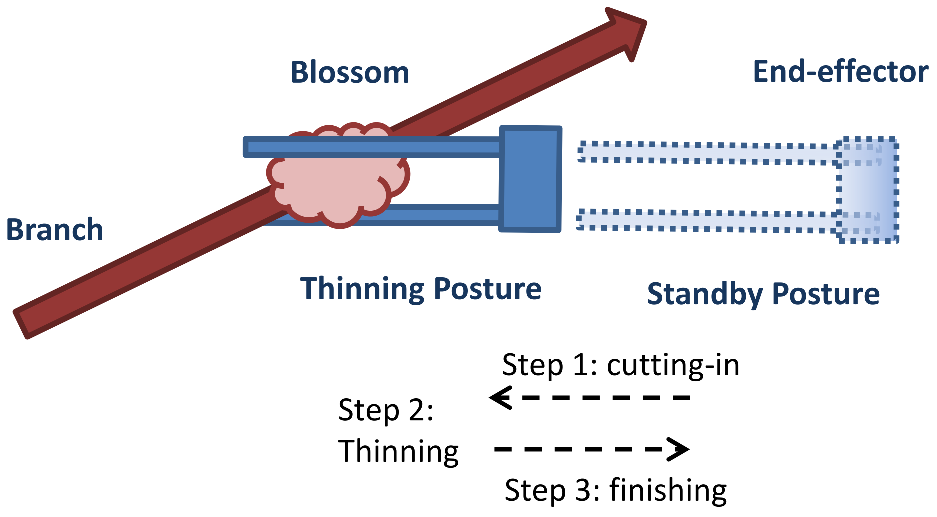


Figure 3.11 Illustration of the end-effector approaching the target blossom and retreating.

The robust design of the end-effector as shown here does not require high precision in the blossom location, as its brushes remove blossoms irrespective of their orientation along the branch circumference (Figure 3.12a). In addition, the length of the two forks provides a larger “strike zone” as shown in Figure 3.12b, so accurate placement is not critical.

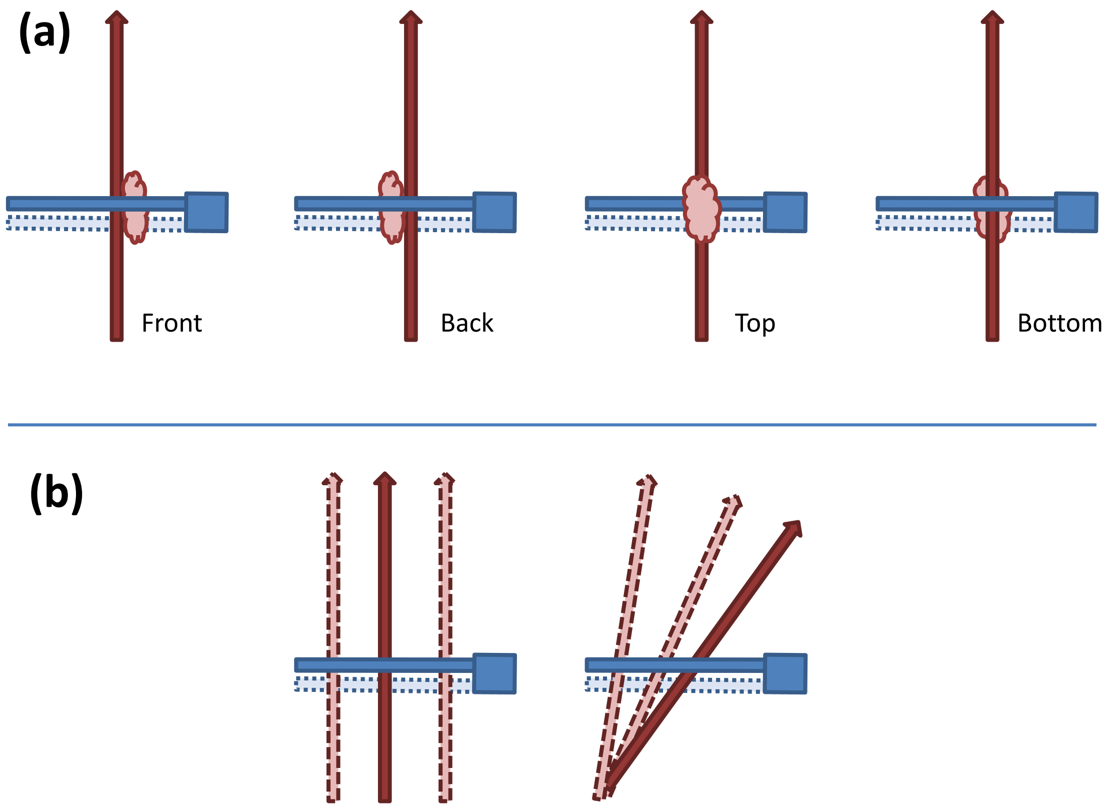


Figure 3.12 Top view of the relative location between the end-effector and the branches. Only the top fork (solid-rod) of the end-effector is visible from this view angle, the bottom one (dotted-rod) is provided for clarity. (a) The orientation of the blossom is not critical to the end-effector. (b) The length of the forks yields a larger strike zone to the end-effector for thinning.

3.3.2 Representative End-effector

Because the size of the full scale end-effector was too large for the scaled-down model tree, the two forks of the end-effector were represented by a scaffold made from electric wire (Figure 3.13). Since no end-effector was present, the arrival of the forks at the blossom location was verified by projecting visible laser light (Figure 3.14a) onto the target coordinates.

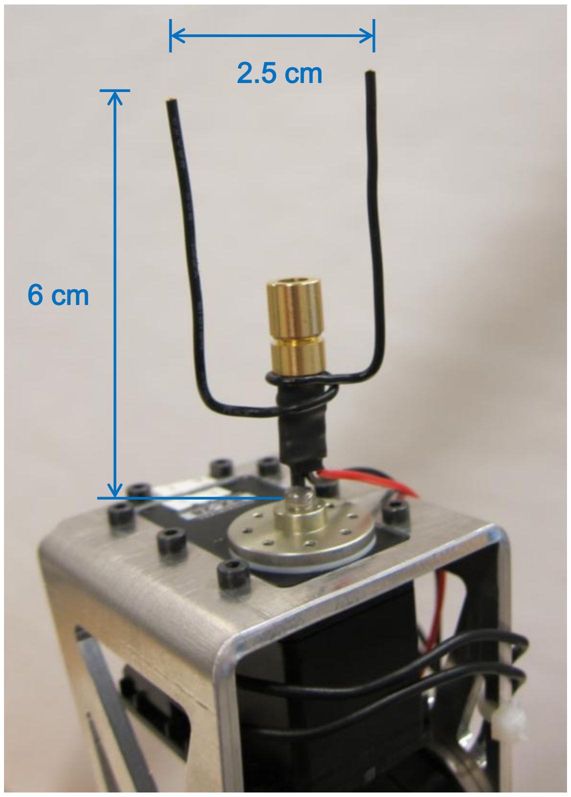


Figure 3.13 Representative end-effector comprising a laser module and a scaffold representing the two forks in the full-scale end-effector.

The laser module had a dimension of 8x13mm and comprised a 5 mW laser diode of 405 nm wavelength (blue), a built-in driver circuit, and a housing with collimating lens (AixiZ Service & International LLC, Houston, Texas). To command the module through the main program written in MATLAB®, an electronic switch was built and actuated by a DAQ device (NI USB-6008, National Instruments®, Austin, Texas, Figure 3.14b). One of its output ports was responsible for sending a signal from 0 to 5 V to control a circuit of transistor. Although the voltage requirement of the laser module was 5 V which was equal to the maximum output voltage of the USB-6008 ports, the voltage at the output port dropped when actuating the laser module directly. This issue was addressed by applying a PNP transistor that drives the laser module as shown in Figure 3.15, which was drawn using CAD software PSpice (Cadence Design Systems, Inc, San Jose, California). When 5 V was applied, the laser module was off; while 0 V was applied, the laser module was on.

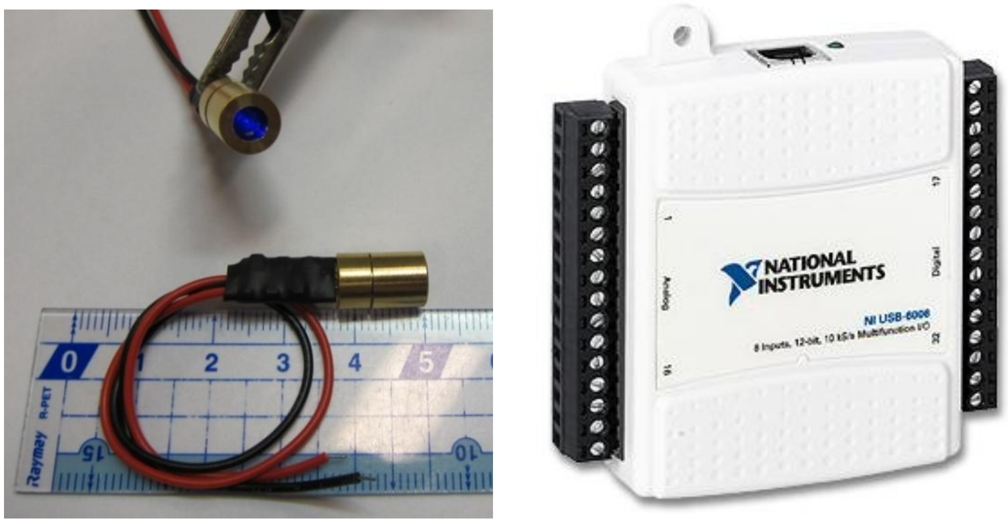


Figure 3.14 Left: Blue laser module by AixiZ. Right: DAQ device NI USB-6008 (Courtesy of National Instruments®).

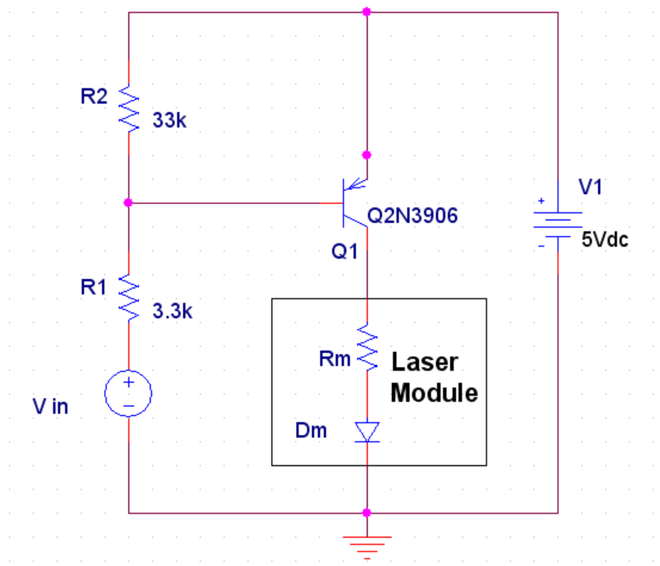


Figure 3.15 Circuit of an electronic switch for the laser module. The values of *Rm* and *Dm* are not listed by the manufacturer.

3.4 Task Definition

As shown in Figure 1.2, a heuristic algorithm, which considered the horticultural practice of thinning, was developed by PSU and would determine which blossoms should be removed and which should be retained. The function of the task planning component is shown in Figure 3.16. Having the coordinates of the target blossoms and the orientations of the branches, the task definition component decides what posture the manipulator should use for thinning. A posture is composed of a coordinate vector (3-by-1) and an orientation matrix (3-by-3) for the end-effector. Two task definition methods, the fixed z-axis method (section 3.4.1) and the floating z-axis method (section 3.4.2), were developed in this study, and the latter requires the current posture of the end-effector as an additional input argument.

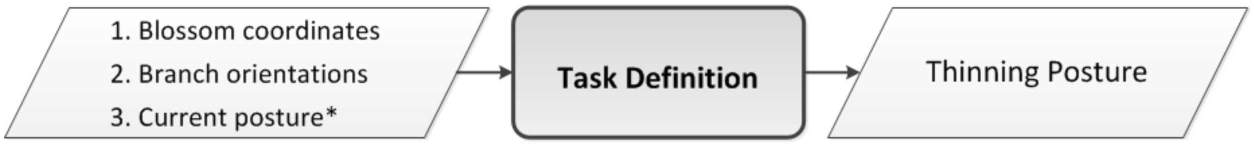


Figure 3.16 Input and output of the task definition unit.

3.4.1 Fixed Z-Axis Method

A good thinning posture required the two forks of the end-effector to go across the branch, in other words, the two forks should be aligned to the normal of the orientation of the branch as shown in Figure 3.17. In a three-dimensional space, the normal of a vector is not unique; instead, an infinite number of candidates form a normal plane perpendicular to the orientation of the branch (Figure 3.18). As shown in Figure 3.12b, the end-effector was not sensitive to the depth changes of the branch, and hence it assumed that the branch lies on a two-dimensional plane as shown in Figure 3.17. A practical solution would imply having the z-axis of the end-effector (tool frame), which is the direction of the forks, always pointing along the x-axis of the world frame. To finish the placement, one more axis of the tool frame should be assigned. By aligning the x-axis of the tool frame with the normal vector of the branch at the blossom location, the orientation of the end-effector can be determined.

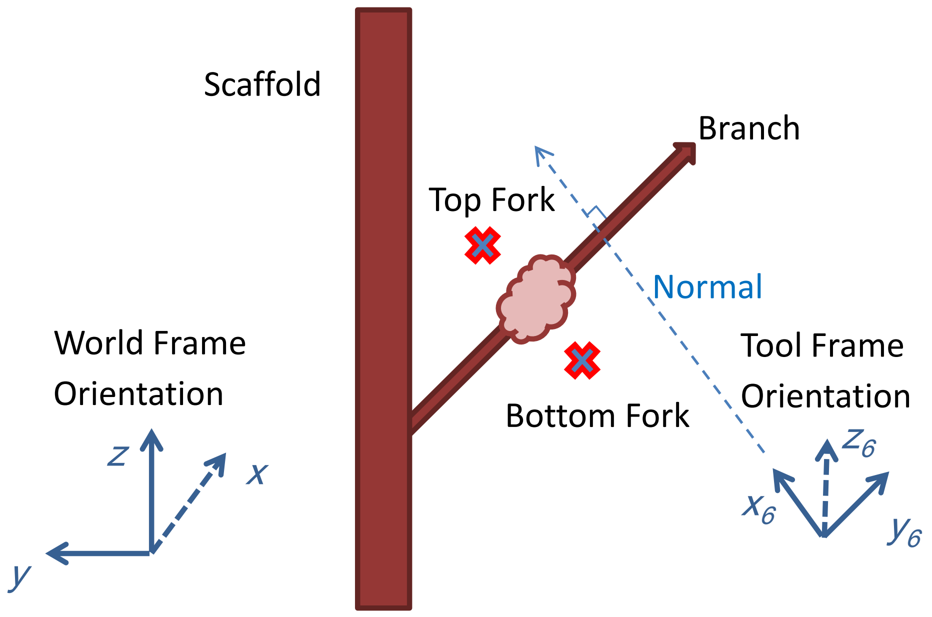


Figure 3.17 A simplistic 2D model illustrates how placing the end-effector at the thinning position was accomplished by aligning the tool frame.

Before placing the end-effector at the target to reach its goal configuration, the manipulator entered a standby position, which was at a 4-cm distance from the target position along the z-axis of the tool frame as shown in Figure 3.11. In effect, the possibility of a collision between the end-effector and the branch would decrease because the robot was moving at a safe distance when traveling from one target to the next.

3.4.2 Floating Z-Axis Method

The fixed z-axis method applies in a simplified world, which was not always the case. Thus, a floating z-axis method was developed for situations with more variation. Figure 3.18 shows under which conditions the floating z-axis method becomes imperative. The fixed z-axis method cannot move the manipulator to the marked target blossom because the location of the target was close to the boundary of the manipulator's operational space. With the fixed z-axis method, the operational space was actually decreased due to a stricter limitation on the postures. As mentioned earlier, any potential posture must have the z-axis of the tool frame aligned with the normal plane as displayed. Thus, the floating z-axis method was designed to search this plane and find the optimal solution.

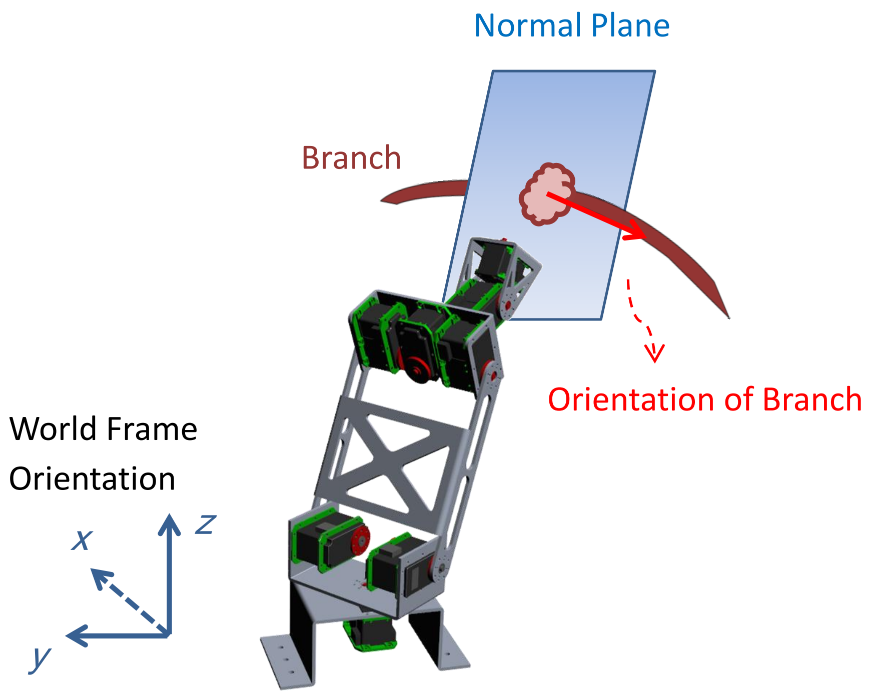


Figure 3.18 The fixed z-axis method does not work for targets close to the boundary of the operational space of the manipulator.

In Figure 3.19, all orange vectors lie on the same normal plane. Since they all point to the target blossom, they were all potential solutions. However, not all of them were feasible for the manipulator due to a finite operational space. After the examination of feasibility, there might be more than one candidate left and a final arbitration is needed. The first step in the arbitration process was scouting the normal plane and sampling all qualified vectors. The second was to test the feasibility of the candidate vectors by solving for their inverse kinematics solutions. Finally, the most appropriate candidate would be selected by measuring the distance between the current configuration and for feasible vectors.

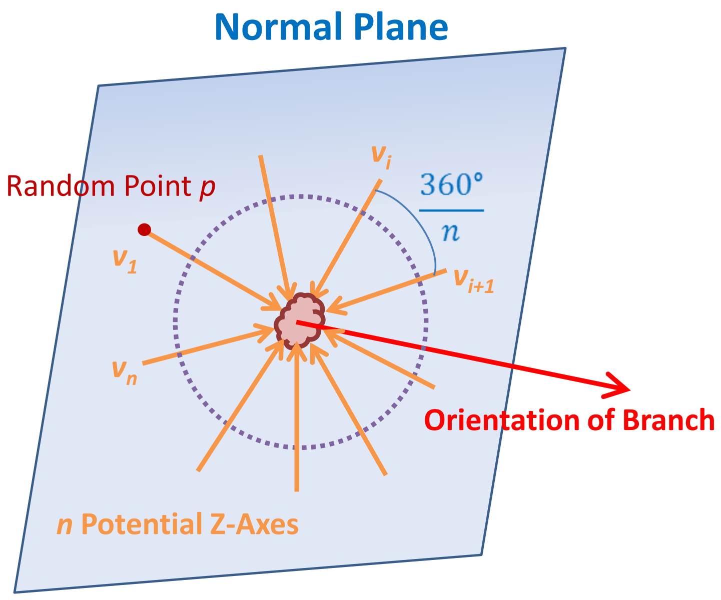


Figure 3.19 Possible solutions (orange vectors) of the tool frame lie on the normal plane of the branch.

Including all the orange vectors, there are infinite potential vectors in the normal plane (Figure 3.19), and hence enumeration is not a possible strategy. A practical method is sampling the normal plane by evenly dividing the dotted circle. The initial step is randomly picking a point *p* on the normal plane, where *p* and the target blossom form a vector v_1. If v_1 was chosen as the final solution, the final orientation for the tool frame would be R_1, which is a 3-by-3 matrix. This matrix could also be expressed as $[r_x, r_y, r_z]$. The r_y was simply the 3-by-1 orientation vector of the branch, the r_z was identical to v_1, and r_x could be decided by the cross product from r_y to r_z. However, R_1 was not always the solution. Instead, the final candidate can be in the range R_1, R_2, ..., R_n, which represent *n* corresponding rotation matrices to *n* vectors on the dotted circle in Figure 3.19. All these vectors have a uniform angle difference θ, which is $(360/n)$ ° about the branch orientation vector r_y. Note that all the potential rotation matrices have the same r_y, and hence there exists a relationship as follows:

$$R_{i+1} = R_i R_\theta = R_i \begin{bmatrix} \mathrm{c}_\theta & 0 & \mathrm{s}_\theta \\ 0 & 1 & 0 \\ -\mathrm{s}_\theta & 0 & \mathrm{c}_\theta \end{bmatrix} \qquad (2)$$

In equation 2, R_{i+1} can be acquired by simply rotating R_i through θ around the branch vector, and this operation can be completed by multiplying R_i by a transformation matrix R_θ for rotation. Since R_1 is solved in the beginning, the R_2, R_3, ..., R_n can be solved successively.

After sampling, there would be *n* rotation matrices R_1, R_2, ..., R_n. By sending the pairs of a rotation matrix and coordinates of target blossoms to the inverse kinematics calculator, which is described in the next section, the feasibility of these candidates can be examined. The calculator outputs travel angles for each joint. If any of the angles is out of the predefined range, the corresponding candidate was labeled as unfeasible and ignored in the next step.

Generally, there are multiple feasible solutions left after the inverse kinematics calculation. Therefore, a criterion must be applied to ensure the uniqueness of the final answer. The idea of “minimum effort” as a criterion was considered here. The movement from the current posture to the next posture should be minimized because in the real world, intuition favors shortcuts which typically save energy and time. And hence, the criterion was stated as follows:

$$\text{distance}(q_0, q_f) = \min_{k\in[1,n]} \sum_{i=1}^{6} \left| q_{0,i} - q_{k,i} \right|$$

Where the distances from the current configuration q_0, which is composed of six joint variables θ_1, θ_2, ..., and θ_6, to all the feasible configurations q_k are calculated first; subsequently, the posture yielding the minimum value was taken as the final answer, q_f. The proposed floating z-axis method is summarized in a flowchart as shown in Figure 3.20. Like the fixed z-axis method, the standby positions are obtained by retreating 4 cm away from the thinning position along the z-axis of the tool frame. The standby postures rather than the thinning postures are regarded as the “current configurations” when evaluating the distances between q_0 and q_k.

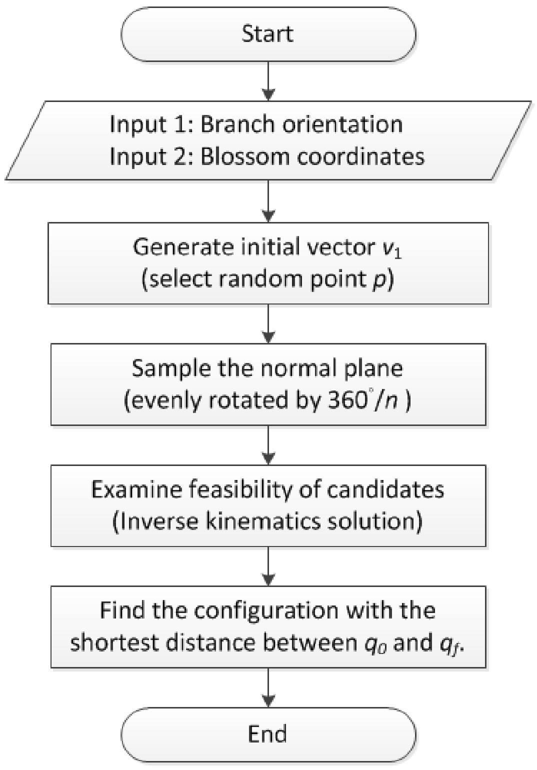

Start
Input 1: Branch orientation
Input 2: Blossom coordinates
Generate initial vector v_1
(select random point p)
Sample the normal plane
(evenly rotated by $360^{\circ}/n$)
Examine feasibility of candidates
(Inverse kinematics solution)
Find the configuration with the
shortest distance between q_0 and q_f.
End

Figure 3.20 Flowchart of the floating z-axis method for task definition.

3.5 Inverse Kinematics Calculation

In the working space, the location of the end-effector was described by a set of x-y-z coordinates and an orientation vector; they could be expressed as a 3-by-1 position vector o_{ef} (the same as o_6^0) with a 3-by-3 orientation matrix R_{ef} (identical to R_6^0). The numbering in the superscripts and subscripts refers to the frame system as shown in Figure 3.8. To move the manipulator, the input geometrical information o_{ef} and R_{ef} were described in a Cartesian coordinate system. However, to actuate each joint, the information described in configuration space was required which means the revolving degree of each motor should be specified. The calculation of an inverse kinematics solution (IKS) was responsible for bridging the transformation process.

Generally, finding solutions to this problem was difficult. Fortunately, solutions to six-DOF manipulators with the last three joints configured as a spherical wrist can be found by a strategy called kinematics decoupling. The strategy decouples the problem into two smaller problems, the inverse position kinematics and the inverse orientation kinematics. The solutions are listed as follows in advance, and the definition of symbols and detailed step-by-step calculation are explained in later subsections.

$$\theta_1 = \text{atan2}(y_c, x_c)$$

$$\theta_2 = \text{atan2}(s, r) - \text{atan2}(a_3 s_3, a_2 + d_4 c_3) \qquad \text{with } r = \sqrt{x_c{}^2 + y_c{}^2} - a_1,\ s = z_c - d_1$$

$$\theta_3 = \text{atan2}(\pm\sqrt{1 - D^2}, D) \qquad \text{with } D = (r^2 + s^2 - a_2{}^2 - d_4{}^2) / (2 a_2 d_4)$$

$$\theta_4 = \text{atan2}(r_{23}, r_{13}) \ \ or \ \ \text{atan2}(-r_{23}, -r_{13})$$

$$\theta_5 = \text{atan2}(\pm\sqrt{1 - r_{33}{}^2}, r_{33})$$

$$\theta_6 = \text{atan2}(r_{32}, -r_{31}) \ \ or \ \ \text{atan2}(-r_{32}, r_{31})$$

In the above solution, the matrix element r_{ij} comes from R_6^3or T_6^3 rather than the T_6^0 mentioned earlier. Note that the definitions of *atan2* adopted in this thesis is *atan2(y, x)*, which agrees with MATLAB®, instead of the *atan2(x, y)* accepted by Spong et al. (2006).

3.5.1 Inverse Position Kinematics

The first step in kinematics decoupling was recovering the position of the wrist center o_c

from the input argument o_{ef} as follows.

$$o_c = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = o_{ef} - d_3 R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Secondly, a geometric approach was applied for solving the first three joint variables, which comprised an elbow manipulator. By projecting this elbow manipulator onto 2D planes, which are x_0-y_0 plane and x_0-z_0 plane, the problem becomes a straightforward trigonometry exercise.

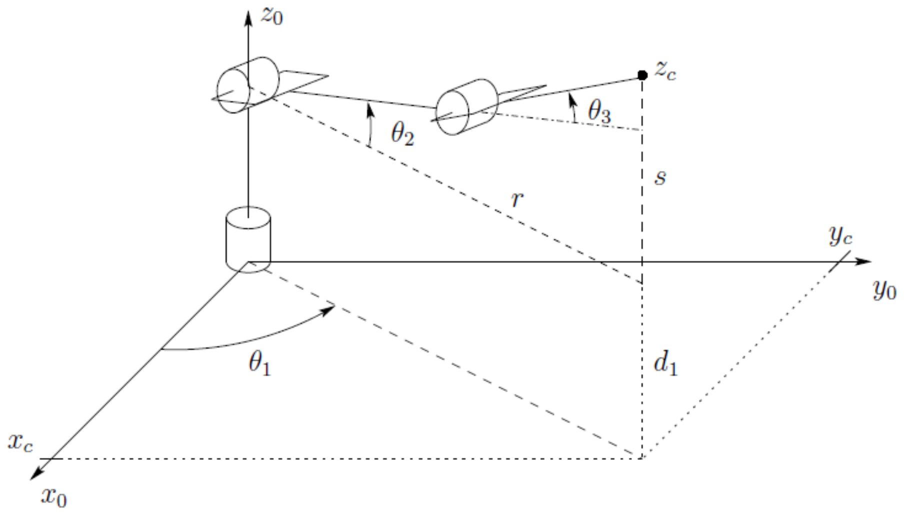


Figure 3.21 Elbow manipulator with frame 0 (Spong et al., 2006).

In Figure 3.21, the x_0 axis and the projection of the manipulator on the x_0-y_0 plane has an angle θ_1 or:

$$\theta_1 = \text{atan2}(y_c, x_c)$$

In fact, there can be another solution that represents a leaning-backward posture with the wrist center very close to the z_0-axis.

$$\theta_1 = \pi + \text{atan2}(y_c, x_c)$$

Considering that the second answer was not helpful for blossom thinning, the control software ignored this solution automatically; thus, the uniqueness of θ_1 was ensured.

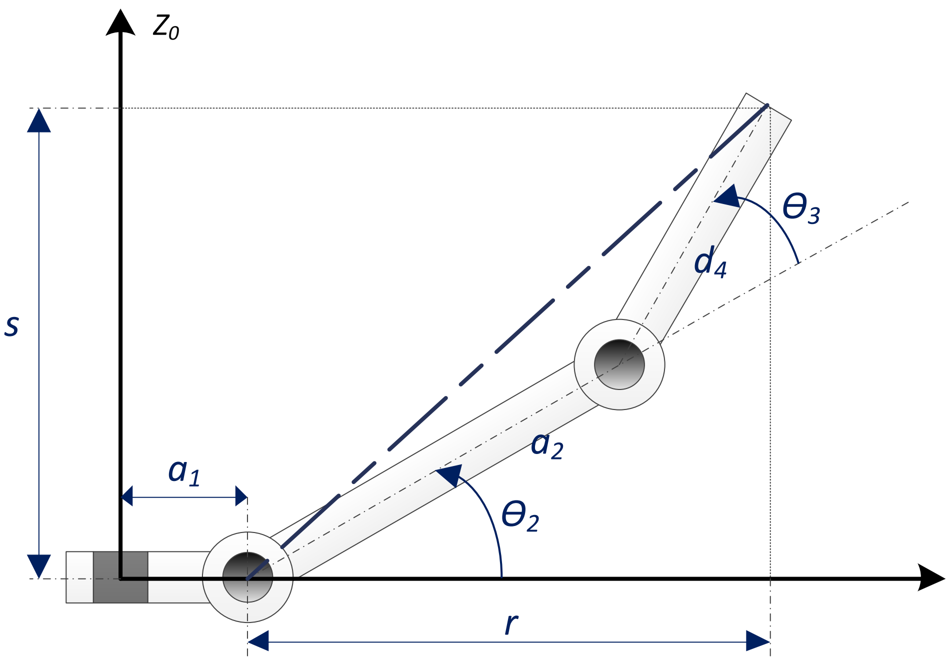


Figure 3.22 Projection plane of links 1, 2 and 3.

Projecting links 1, 2 and 3 and z_0-axis on to a plane simplified the problem. As shown in Figure 3.22, θ_3 could be solved by applying the law of cosines. Note that the horizontal axis was not necessary to be x_0 or y_0. Assigning $r = \sqrt{x_c^2 + y_c^2} - a_1$ and $s = z_c - d_1$, here yielded:

$$\cos(\pi - \theta_3) = \frac{a_2^2 + d_4^2 - (r^2 + s^2)}{2a_2 d_4}$$

$$\cos\theta_3 = \frac{(r^2 + s^2) - a_2^2 - d_4^2}{2a_2 d_4} \equiv D$$

$$\theta_3 = \text{atan2}(\pm\sqrt{1-D^2}, D)$$

The positive and negative solutions for *θ_3* represent the elbow-down posture and the elbow-up posture respectively. It is obvious that there are two solutions for *θ_2* corresponding to each *θ_3* respectively.

$$\theta_2 = \text{atan2}(s, r) - \text{atan2}(a_3 s_3, a_2 + d_4 c_3)$$

3.5.2 Inverse Orientation Kinematics

The first three joint variables *θ_1*, *θ_2*, and *θ_3* determined the rotation matrix R_3^0 and transferred the reference frame from $o_0x_0y_0z_0$ to $o_3x_3y_3z_3$, which was the frame of the wrist center. Since R (input) and R_3^0 were both known, the R_6^0 was solved as follows.

$$R = R_0^6 = R_3^0 R_6^3$$

$$R_6^3 = (R_3^0)^{-1} R = (R_3^0)^T R = \text{known values}$$

In addition, *R_6^3* can be represented by *θ_4*, *θ_5*, and *θ_6* as

$$R_6^3 = \begin{bmatrix} c_4c_5c_6 - s_4s_6 & -c_4c_5s_6 - s_4c_6 & c_4s_5 \\ s_4c_5c_6 + c_4s_6 & -s_4c_5s_6 + c_4c_6 & s_4s_5 \\ -s_5c_6 & s_5s_6 & c_5 \end{bmatrix} = \text{ expression in } \theta_4,\ \theta_5,\ \text{and } \theta_6$$

Where *c_4* represents *$\cos\theta_4$*, etc. The above form agrees with a specific form of rotation called Euler angle transformation (Spong et al., 2006). To find a solution to it, the following two cases must be considered.

Case 1: Not both *r_{13}* and *r_{23}* of R_6^3 are zero.

If the condition, $r_{33} \neq 1$ or -1. Thus, $c_5 = r_{33}$, and $s_5 = \pm\sqrt{1 - r_{33}^{\ 2}}$.

$$\theta_5 = \text{atan2}(\pm\sqrt{1 - r_{33}^{\ 2}}, r_{33})$$

With $s_5 > 0$, θ_4 and θ_6 can be determined by

$$\theta_4 = \text{atan2}(r_{23}, r_{13})$$

$$\theta_6 = \text{atan2}(r_{32}, -r_{31})$$

Similarly, with $s_5 < 0$

$$\theta_4 = \text{atan2}(-r_{23}, -r_{13})$$

$$\theta_6 = \text{atan2}(-r_{32}, r_{31})$$

<u>Case 2:</u> Both r_{13} and r_{23} are zero.

Since R_6^3 is a rotation matrix, orthogonality implies $r_{33} = 1$ or -1 with $r_{31} = r_{32} = 0$.

$$R_6^{\ 3} = \begin{bmatrix} r_{11} & r_{12} & 0 \\ r_{21} & r_{22} & 0 \\ 0 & 0 & \pm 1 \end{bmatrix}$$

If $r_{33} = 1$, then θ_5 is 0 and the equation becomes

$$R_6^{\ 3} = \begin{bmatrix} c_4c_6 - s_4s_6 & -c_4s_6 - s_4c_6 & 0 \\ s_4c_6 + c_4s_6 & -s_4s_6 + c_4c_6 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c_{4+6} & -s_{4+6} & 0 \\ s_{4+6} & c_{4+6} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Where the sum of $\theta_4 + \theta_6$ can be solved by

$$\theta_4 + \theta_6 = \text{atan2}(r_{21}, r_{11}) = \text{atan2}(-r_{12}, r_{11})$$

Although the sum can be determined, there can be infinite combinations of θ_4 and θ_6. To address this issue, θ_4 was set to 0 in the control software. This is reasonable, since this choice minimizes the movement of the manipulator. Similarly, if $r_{33} = -1$, then θ_5 is 180°. Thus,

$$R_6^{\ 3} = \begin{bmatrix} -c_4c_6 - s_4s_6 & -c_4s_6 - s_4c_6 & 0 \\ -s_4c_6 + c_4s_6 & -s_4s_6 + c_4c_6 & 0 \\ 0 & 0 & -1 \end{bmatrix} = \begin{bmatrix} -c_{4-6} & -s_{4-6} & 0 \\ -s_{4-6} & c_{4-6} & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

And the solution becomes

$$\theta_4 - \theta_6 = \text{atan2}(-r_{12}, -r_{11})$$

Where the default value of θ_4 was set to 180°.

3.5.3 Uniqueness and Solution Set

The previous two subsections indicate that there is no unique solution to any input. In fact, all valid inputs fall into one of three scenarios being four solutions, two solutions, and infinitely many solutions. In the most common scenario, which is the four-solution case, there are two possible θ_3 with two θ_2, two θ_4, two θ_6, and four θ_5 accordingly; their potential combinations are shown in Table 3.3, where I, II, III, and IV are the number of possible values. The control software utilized the same layout to store its working variables. The reason why the number of combinations is not $1\times2\times2\times2\times4\times2=64$ is that some joint variables are dependent upon others; for example θ_2 is derived from θ_3 as mentioned earlier.

Table 3.3 Potential combinations of all possible values of joint variables.

Joint Variable (Num. of candidates)		θ_1 (1)	θ_2 (2)	θ_3 (2)	θ_4 (2)	θ_5 (4)	θ_6 (2)
Solution	1	I	I	I	I	I	I
	2	I	I	I	I	II	I
	3	I	II	II	II	III	II
	4	I	II	II	II	IV	II

The non-uniqueness issue can be partially solved by setting the operating range of the motors, which are the actuators of each joint, as listed on Table 3.4. For example, if for a given input the calculation provides two possible values of θ_4, which are 0° or 180°, obviously the latter is not applicable because the range of joint 4 is set to [-150°, 150°]. Hence, one or two solutions can be removed from the list of potential candidates. A second helpful tactic is giving default values or setting preferences for the remaining candidates allowing convergence to a single solution. This is especially important in the infinitely-many-solution scenario. As discussed in the previous subsection, when $\theta_4 + \theta_6 = 180°$, θ_4 is set to 0° for a feasible solution.

Table 3.4 Operating range of motors for each joint.

Joint	1	2	3	4	5	6
Upper limit	135°	99°	90°	150°	91°	150°
Lower limit	-135°	-45°	-99°	-150°	-91°	-150°

Note that the operating ranges for individual motors are higher than the manually-set limits except for joint 4 and 6. The current limits are determined by the design of the robot and assumptions regarding the operation conditions. For example, the operating range for the motors on joint 3 is 300° in total, but the given boundaries only reach 189° (|90°|+|-99°|) as maxima, because some postures of the manipulator are not feasible for blossom thinning, for example, a posture where the end-effector is pointing away from the tree.

4 EXPERIMENTAL DESIGN AND RESULTS

In this chapter, assumptions and experiments of various stages are introduced, starting from a simplistic 2D model (section 4.2) to a more advanced representative case (section 4.3).

4.1 World Frame and Tool Frame Definition

A world frame, also known as a base frame, was the reference coordinate system in which the manipulators and all other objects were positioned. The coordinate system $o_0x_0y_0z_0$ in Figure 3.8 was adopted as the world frame for the experiments. The origin of $o_0x_0y_0z_0$ was assigned to the floor, where the bottom of the manipulator is located. The tool frame is the coordinate frame attached to the end-effector, which was the $o_6x_6y_6z_6$ in Figure 3.8. Without the end-effector, the DH parameter d_6 was 6.4 cm; with the representative end-effector, d_6 extended 6 cm along the z-axis and became 12.4 cm.

4.2 2D Simplistic Model

At this stage, the orientations of the branches were assumed to have no projection onto the x-axis of the world frame. Although in the real world, the branches have depth changes, this is a sound assumption: As shown in Figure 3.12b, the end-effector was not sensitive to depth variations, and hence the experiments at this stage still reflected the features of real situations. In this section, both of the two data sets were constrained to a 2D plane, which was the best case for the fixed z-axis method to place the end-effector at appropriate orientations.

4.2.1 Experimental Arrangement

The first data set was from a branch as shown in Figure 4.1. This branch was horizontal and perpendicular to the scaffold; it pointed to the direction of (0, 1, 0) in the world frame. The target blossoms were generated by the 5-8 pattern with red stickers as targets, while the yellow stickers indicated blossoms that should be retained. There were 7 targets out of 10 blossoms in data set 1. The second dataset (Figure 4.2) comprised a horizontal segment and a leaning segment with a 45° angle between them. Among the 8 blossoms, there were 6 targets in this set; three on the first segment, and the remaining three on the second segment.

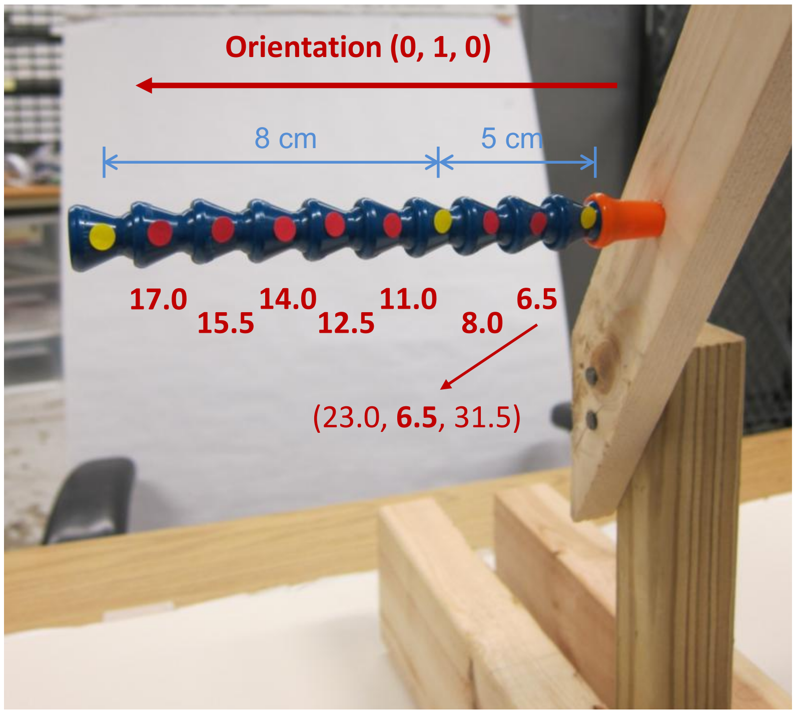


Figure 4.1 Data set 1 comprised a horizontal branch in the Y-Z plane without a projection on the x-axis. The red/yellow stickers represent target/non-target blossoms.

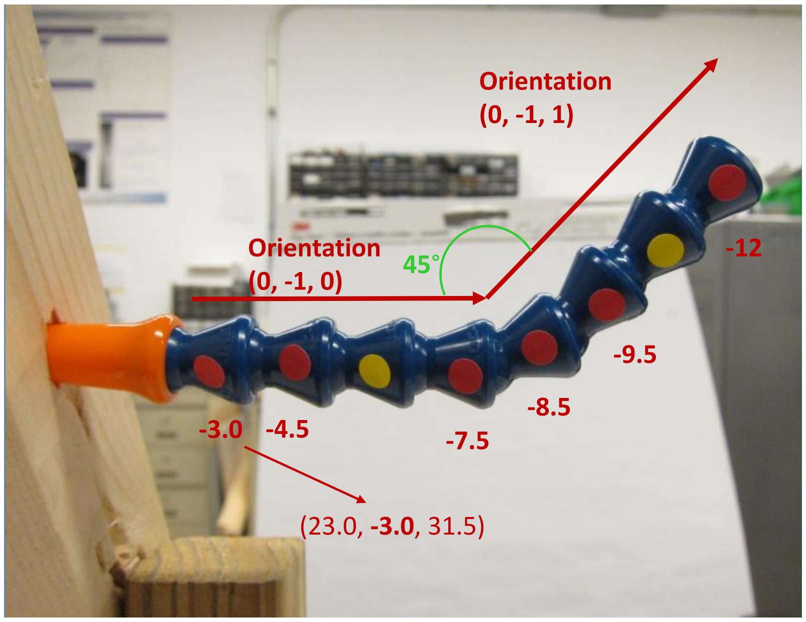


Figure 4.2 Data set 2 included target blossoms on a horizontal segment and a leaning segment.

4.2.2 Results and Discussion

In data set 1, the end-effector reached the first six targets successfully. While trying to reach the 7th target, which has coordinates of (23, 17, 31.5), the end-effector collided with the branch. After the collision, the end-effector still arrived at the target and projected the laser beam onto the sticker. This event demonstrated two things: First, preparing the end-effector at a standby position did not always guarantee a collision-free movement. Second, collision might not necessarily be a grave error. As mentioned earlier, the branches had some flexibility, so a slight collision might not influence the arrival of the end-effector and subsequent removal of blossoms.

To have a closer look at the path to the 7th target, the movement is broken down and displayed in Figure 4.3. The manipulator started from (1), went underneath the branch (2), raised up (3), and finally bumped into the branch while moving forward (4). This was far from the expected straight-line movement. The main reason was that all the motors were actuated at the same speed at the same time; however, their travel distances were different. Thus, unexpected intermediate postures were encountered during the movement. One potential solution was to set several intermediate postures between the standby posture and the thinning posture along the expected path. This results in an irregular movement, since the manipulator starts and stops several times along the path. With delicate velocity control for individual motors, this phenomenon could be improved. Yet, the computing power for the additional inverse kinematics computation to each intermediate posture will slow down the process significantly. Currently each posture required about 0.11 seconds for finding an inverse kinematics solution. Without intermediate postures, every target required 0.22 seconds for computing the thinning posture and standby posture. More intermediate postures would increase the time requirement significantly, for example, ten of them would add 1.1 extra seconds.

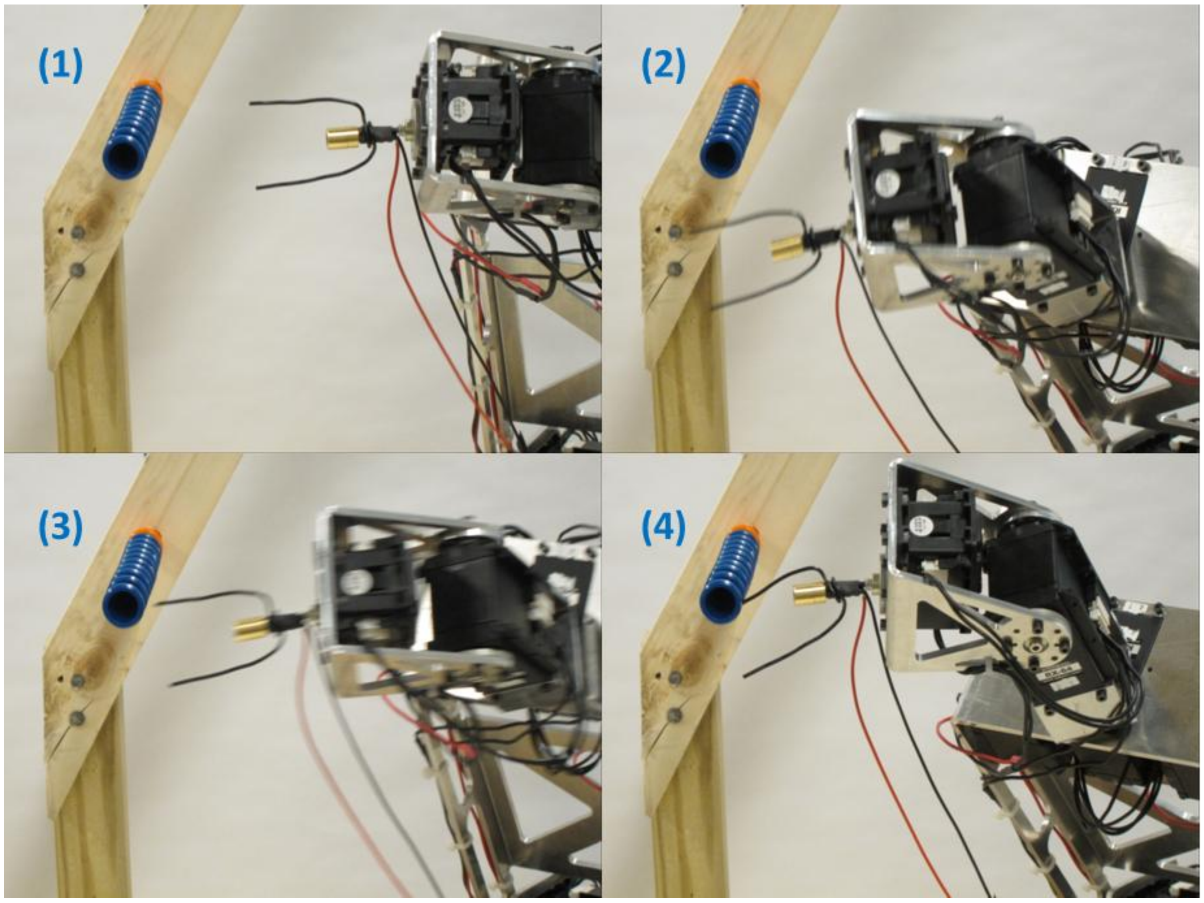


Figure 4.3 Break-down movement for the collision of the 7th target in the data set 1. The path from the standby posture to the thinning posture was not a straight line. The manipulator started from posture (1) and proceeded to posture (4), where the collision occurred.

All the targets in data set 2 were visited smoothly. Figure 4.4 shows that the laser projects light onto the rims of target 1, 3, and 4 rather than at the centers. However, this was not a significant issue since the real blossoms are larger than the stickers. The orientation of the scaffold changed according to the orientation of the branch segment. It indicated that the algorithm calculated the normal correctly and ensured that the forks of the end-effector traversed across the branch completely.

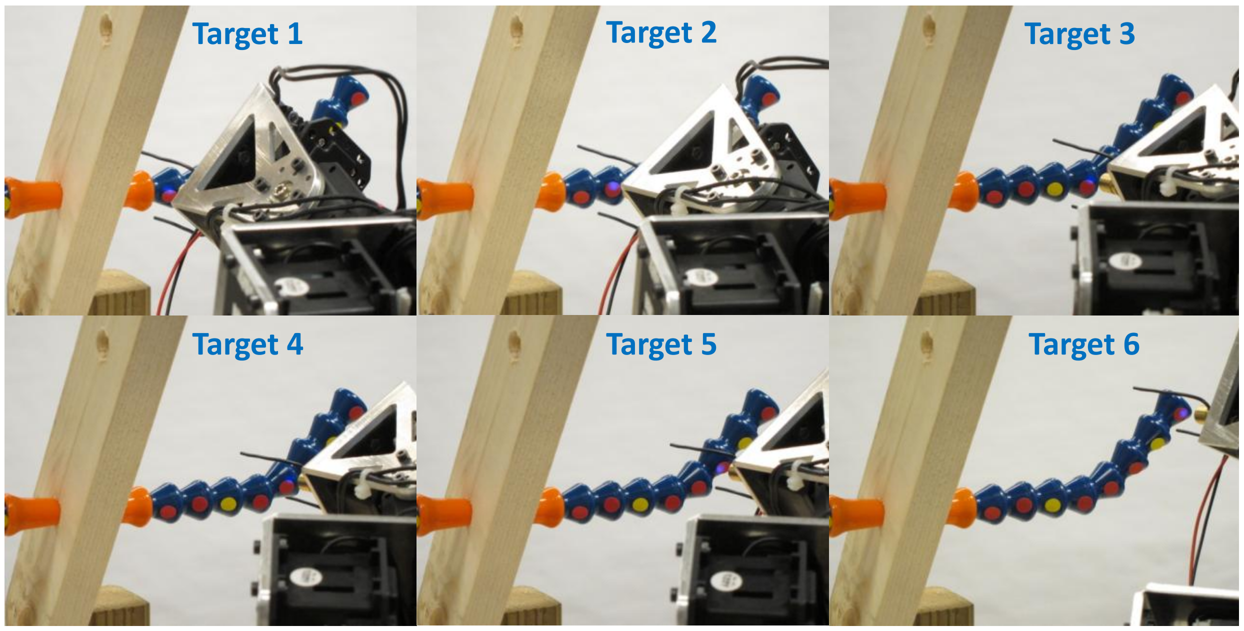


Figure 4.4 Results of data set 2. The orientation of the end-effector changes along the branch.

4.3 3D Model

After the discussion of the two-dimensional simplistic examples, this section discusses a more representative case being a three-dimensional model of the branch. The floating z-axis method was applied here for placing the end-effector at an appropriate orientation. In addition, to ensure the path from the standby posture to the thinning posture remained a straight line, varying amounts of the intermediate points were tested to study their influence.

4.3.1 Experimental Arrangement

Unlike data set 1 and 2, data set 3 was defined in a three-dimensional space (Figure 4.5). This branch contained three segments, which had orientation (0, -1, 1), (-1, 1, 1), and (-1, -1, 0); only the second and third segment were used in this section. All three digits of the coordinates for each target were unique. The increments of the coordinates between targets did not perfectly agree with the orientation vectors. The purpose of this design was to reflect the inaccuracy in a real orchard operation and to test the robustness of the proposed task planning algorithm. The other feature of this example was the “heads up” position of the branch. As displayed in Figure 4.5, the location of the branch was higher than the manipulator, and hence it was not possible to implement the fixed z-axis method. Since its z-axis of the tool frame was always pointing along the x-axis of the world frame, the fixed z-axis method only applied when the manipulator could “overlook” the target. This was the motivation for designing a more universal method termed the floating z-axis method (see subsection 3.4.2).

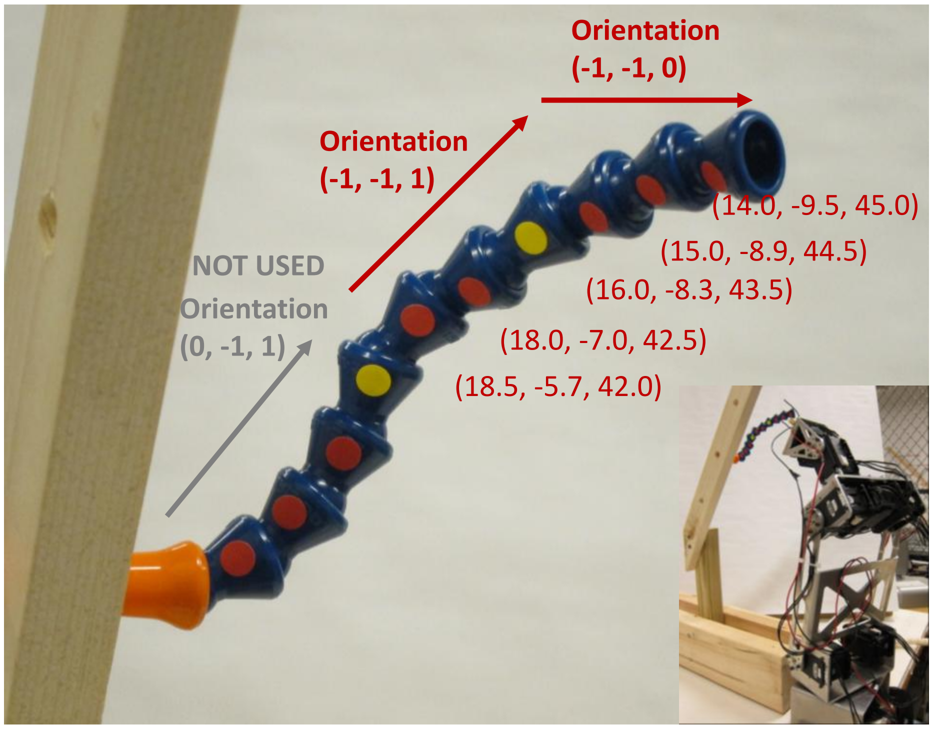


Figure 4.5 Data set 3 is a branch in a three-dimensional space. Compared with data set 1 and 2, this is a more representative example where the fixed z-axis method failed.

4.3.2 Results and Discussion

The floating z-axis successfully determined the thinning posture for the end-effector as shown in Figure 4.6. This method enabled the end-effector to point upward for approaching the “overhead” targets. Since the laser was approaching from the bottom, its illuminating spots were not visible from the side views in Figure 4.6. There was a large movement to transit from target 2 to target 3; nevertheless, from target 1 to target 2 and from target 3 to target 5, the movements were modest. This was expected since the “minimum effort” principle was the central premise of this method. To adequately scout the normal plane (Figure 3.19), *n* was assigned the number 50 and the distance between the solution candidates was 7.2° (360° divided by 50). Under this setting, a posture would require 5.5 seconds (0.11 seconds times 50) to find.

By using *n* of 10 and 20, this method failed frequently, and no feasible posture was found for placing the end-effector. Using a value of 30 or 40, the paths generated by different rounds were visually distinct, and the repeatability of resulting paths was low. Some of the paths were counter intuitive and seemed to take a longer route compared to the intuitive ones. The reason was that this method required the current posture for computing the next posture, and errors accumulated and passed through all the visits that the manipulator made. The maximum error for the initial vector was (360/n)° from the optimal one. Consequently, the higher the *n* value is, the lower the error can be, which increases the repeatability of the resulting paths. Although increasing the *n* value improved resulting paths, the tradeoff between time and performance was expensive. A 5.5-second cost was already significant under the setting of 50; a higher value of *n* might be expensive in terms of time consumption.

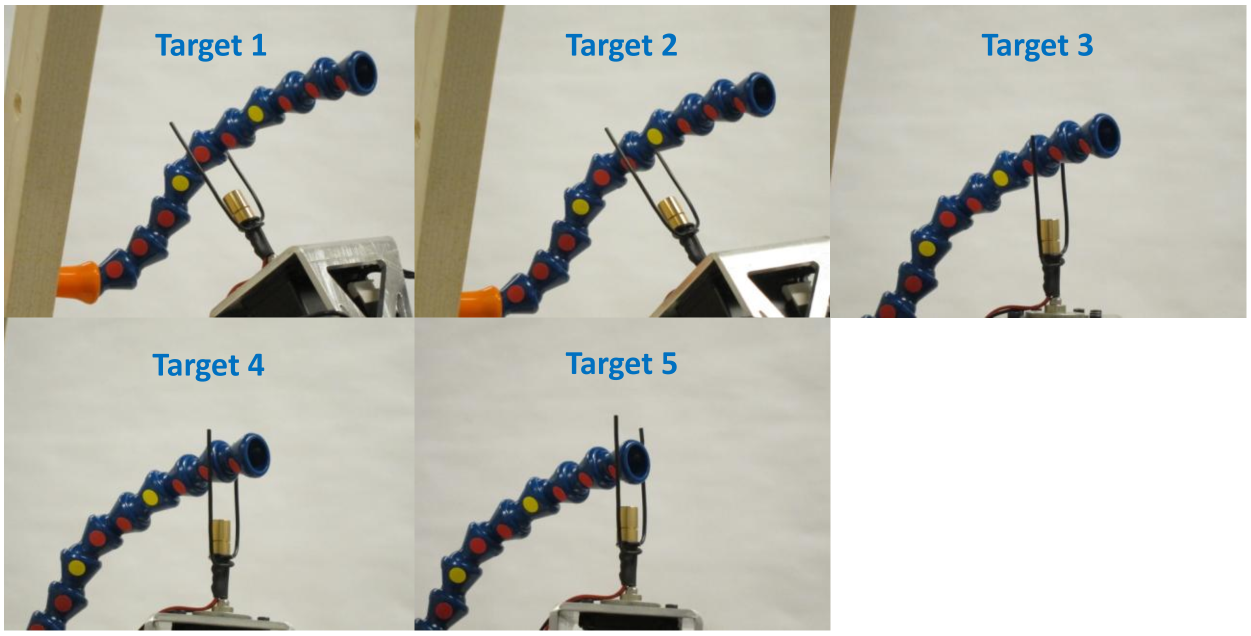


Figure 4.6 Results of data set 3. The floating z-axis method enables the end-effector for pointing upward to the targets.

The floating z-axis method ensured an appropriate final posture for the end-effector, but it did not consider the path that the manipulator was following. Although 4-cm branch distance standby postures were calculated for all targets, the end-effector collided with the branch frequently, in fact, none of the routes were collision-free. The break-down movement in Figure 4.7 showed how the end-effector approached the 1st target in data set 3 without visiting any intermediate points. The end-effector started from the standby posture (1), collided with the branch while moving toward of the branch (2), ascending up to the thinning position (3), and finally collided with the target position from the back of the branch (4). Similar to the collision of the 7th target in data set 1 (Figure 4.3), the demonstrated path resembled an arc rather than a straight line. However, not all collisions in data set 3 can be ignored. Since they deformed the representative end-effector, they must be avoided.

To have a “straighter” cut-in path rather than a curved one, 20 intermediate points (including the standby point) were assigned and evenly distributed from the standby posture to the thinning posture. All twenty points required computing resources for inverse kinematics solutions, and they added 2.2 seconds (0.11 times 20) to the 5.5 seconds for orientation calculation. The total computing time for a single target blossom would be 7.7 seconds (2.2 plus 5.5). Fewer intermediate points being two, five, and ten were also tested. Two and five points improved somewhat but could not avoid most of the collisions; ten points resulted in minor occasional brushes with the branch. When the number of points increased to 20, the path became collision-free.

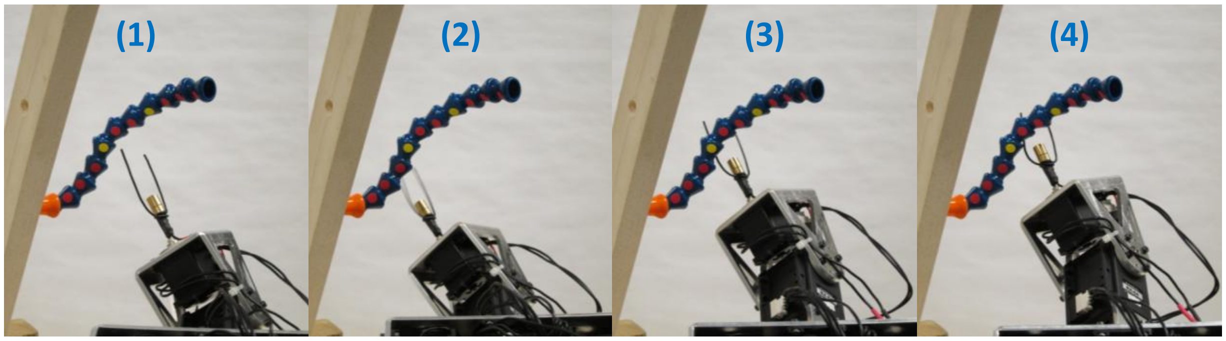


Figure 4.7 Break-down movement of the collision in the data set 3. Collisions take place in frame (2) and frame (4) separately.

5 CONCLUSIONS

The first part of this chapter summarizes the achievements of this study, and addresses the three research objectives listed in section 1.2. The second part collects valuable insights and identifies insufficiencies of this study.

5.1 Summary

A representative model of peach tree was built for experimental use. Abstracting the features of the scaffolds, branches, and blossoms from real trees helped the design of the end-effector and the task planning algorithm. The model tree enabled the proof-of-concept experiment in the laboratory. A custom tabletop manipulator with sufficient payload was built from servo motors for this research. Software written in MATLAB® and a calibration tool written in LabVIEW® were also developed to control the manipulator.

A task planning algorithm for a manipulator was developed for automated selective thinning of peach. For task definition, a rudimentary fixed z-axis method was first tested in two-dimensional simplistic cases without any problems. However, the fixed orientation of the tool frame imposed a strict limitation on the manipulator and greatly reduced its operational space. With the proposed floating z-axis method, the algorithm could automatically decide the best approaching orientation for the end-effector. This method was tested in a more representative three-dimensional case and the end-effector successfully visited all the targets.

To reduce collisions, the end-effector approached a target blossom from a 4-cm-away standby position and proceeded along intermediate points evenly allocated along the z-axis of the tool frame. This arrangement ensured that the path was close to a straight line rather than a curve. In addition, instead of moving among target blossoms, the end-effector traveled from one standby point to another, which reduced the probability of collisions. On average, a target required 7.7 seconds for finding a path toward the next target. In conclusion, this study successfully validated a task planning algorithm for automated selective thinning, and proved the concept using a table-top robot model.

5.2 Future Development

Although this thesis has demonstrated the concept of automated selective thinning in a scaled-down prototype, various issues remain unsolved and need to be addressed. Section 5.2.1 through 5.2.3 list the tasks that should be continued in the laboratory. Sections 5.2.4 provides some suggestions for the integration between the manipulator unit and the computer vision unit. Section 5.2.5 lists some ideas about choosing an industrial-grade manipulator, a carrying vehicle, and other devices for the full-scale test. Section 5.2.6 discusses the changes needed to introduce the robotic system into current orchard systems. Section 5.2.7 discusses methods to expedite the control software, and the improving procedures that could be implemented in the future.

5.2.1 Further Experimental Analysis

The experiments conducted in this research have shown the feasibility of an end-effector to approach target blossoms. However, there was only one branch in each data set. More experiments should be conducted to analyze the scenarios of multiple branches as shown in

Figure 5.1. In these cases, other branches become obstacles for the target branch and set constraints for the manipulator's movement. Consequently, locating all obstacles and planning a collision-free path will be crucial. It is clear that applying robotic technologies in orchards cannot take place without adapting the orchard itself to the robotic intervention. For instance, proper pruning with object avoidance in mind may be helpful. Further experimental analyses should draw conclusions regarding what can be done at the engineering side, and what should be done at the horticultural side.

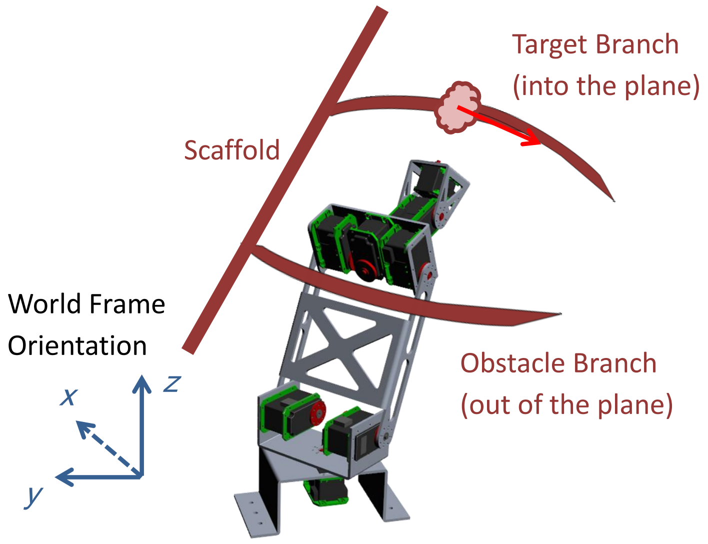


Figure 5.1 Scenario of multiple branches that might become obstacles.

5.2.2 Extension of the Floating Z-Axis Method

The floating z-axis method achieved satisfying results in this study. However, there is ample room for improvement. In subsection 3.4.2, the criterion for "minimum effort" was merely the difference of joint variables in travel angles. A potential flaw behind this method is that all joints are equally weighed during the calculation, which is not fully convincing. Considering the torque involved, moving the joint 1 (waist) of a manipulator requires more energy compared to moving joint 6 (wrist) at the same angular speed for identical travel angles. A dedicated weighing function will make this method more robust and efficient.

5.2.3 Path Planner and Trajectory Planner

The task planning algorithm proposed in this method does not have a well-defined “path planner” or “trajectory planner.” By assigning the 4-cm-away standby point, the manipulator was intentionally kept away from the branch being the potential obstacle. However, in cases such as shown in Figure 5.1, this method might become invalid in a more complicated environment. Thus, a path planner that finds a feasible path would be essential. To do so, a camera mounted on the end-effector might be necessary for enhanced perception of the surrounding environment. Currently, the manipulator moves at the same speed for all joints at all times. However, the manipulator does not need to move slowly while traveling at a distance from the potential obstacles. A trajectory planner can provide a more targeted velocity and acceleration control and reduce the time of operation. One important factor that should be considered is the computing time of these planners. A balance between performance and time consumption should be a central concern in the design process.

5.2.4 Computer Vision Unit

In this thesis, all the test data of target blossoms were artificial rather than measured by a computer vision unit. The vision team of ITTF project has achieved reasonable accuracy in determining blossom locations (Nielsen et al., 2012), but their method still suffers from occlusion (low depth visibility) and limitation to only night time operation to allow controlled lighting. A possible solution is adding an additional camera and light source for tuning. A combination of a fixed camera and a wrist-mounted camera can benefit the collision-free path planning and the placement of end-effector. If higher accuracy is required, it is also possible to equip the robotic thinning system with a vision-system-driven servo control unit, the so-called visual servo control, as discussed by Spong et al. (2006) and Hutchinson et al. (1996).

5.2.5 Selection of Carrying Vehicle and the Manipulator

To travel in the orchard and visit all trees, there is a need for a moving platform to carry the robotic thinning system. The platform can either be a trailer attached to a tractor (Figure 2.7), or an unmanned vehicle such as shown in Figure 2.3. The platform must have an elevating device to change the height of the manipulator since the tree can have a height of 4 m. Considering the weight of industrial-grade manipulators, the elevator might be actuated hydraulically. The moving platform will at least carry the manipulator, robot controller with computer, vision unit, and elevating unit. Extra electric DC power source or transformers might increase the total weight as well. It might be helpful to identify potential carriers in advance, and take their specifications into account while choosing the other components mentioned above.

Finding or developing the ideal robotic arm for thinning operations is essential. Although there are some huge manipulators in the market, it is impractical to deploy them due to the overinvestment in payload and their high weight. For example the FANUC M-2000iA/900L robot has a vertical stroke of 6.2 m and a horizontal stroke of 8.2 m, but its 900 kg payload is excessive with an extreme weight of 9600 kg in total. An in-depth analysis must be conducted to specify the manipulator range, speed requirements, and payload. Since the total weight of all the instruments must not exceed the payload of the moving platform, a compromise between various requirements is expected. An option that might be considered is the possibility to conduct various tasks such as thinning, harvesting, spraying and others.

5.2.6 Adaptation of the Orchard System

Adaptation of the production system is common in the history of agricultural automation and mechanization, and the same is needed in orchards where changes such as tree spacing and pruning methods need to be made to the current production system to allow robotic operations. To accomplish this robot-friendly orchard, close collaboration between agricultural engineers and horticulturalists is essential.

5.2.7 Software Implementation

The speed of the current control software was far from satisfying. By using the floating z-axis method and a standby posture with 20 intermediate points, the software required about 7.7 seconds for the path calculation. Although upgraded hardware could be helpful, radical improvements would be more influential. Some possible solutions are provided here:

Multi-threaded computing. Modern computers are usually equipped with multiple processors or cores in the CPU, but MATLAB® only uses one by default. The end-effector was set to stay in the thinning posture for 5 seconds, because the brushes were assumed to use this time for blossom removal, during which MATLAB® idles. By using a specialized software structure, one processor could keep calculating the inverse kinematics solutions and store them in a first-in-first-out queue. At the same time, other processor could request the queue and command the manipulator.

Dedicated calculation strategy. Solving intermediate postures contributes significantly to the total calculation time. However, since intermediate postures are close together, the distances among them are short, which would imply that the IKS can be computed in an inexpensive manner. Besides, more advanced control theory might be helpful to smoothen the current irregular motion on the cut-in or finishing path for thinning.

Third-party software. Completely customized software made structure modification and concept verification relatively easy. It did benefit this study significantly, but it did not utilize much of the knowledge from the open world. In future deployment, it would be proper to exhaustively survey the available software resources and take advantage of them. The Robotics Toolbox, which is implemented on MATLAB® and maintained by Corke (2011), might be a good starting point.

REFERENCES

Baeten, J., K. Donné, S. Boedrij, W. Beckers, and E. Claesen. 2007. Autonomous fruit picking machine: A robotic apple harvester. In *6th Intl. Conf. on Field and Service Robotics*, 531-539. Chamonix, France: Springer.

Bassi, D., and R. Monet. 2008. Botany and Taxonomy. In *The Peach: Botany, Production and Uses*, 1-36. D. R. Layne, and D. Bassi, eds. Cambridge, MA: CABI Publishing.

Bergerman, M., S. Singh, and B. Hamner. 2012. Results with Autonomous Vehicles Operating in Specialty Crops. In *2012 IEEE Intl. Conf. on Robotics and Automation*, 1829-1835. Saint Paul, MN: IEEE.

Corke, P. 2011. *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*. Berlin, Germany: Springer.

De Koning, C., and J. Rodenburg. 2004. Automatic milking: state of the art in Europe and North America. In *Automatic milking: a better understanding*, 27-37. A. Meijering, and H. Hogeveen, eds. The Netherlands: Wageningen Academic Pub.

DeJong, T. M., K. R. Day, J. F. Doyle, and R. S. Johnson. 1994. The Kearney Agricultural Center perpendicular "V" (KAC-V) orchard system for peaches and nectarines. *Horttechnology* 4(4): 362-367.

Edan, Y., T. Flash, U. M. Peiper, I. Shmulevich, and Y. Sarig. 1991. Near-minimum-time task planning for fruit-picking robots. *IEEE Trans. on Robotics and Automation* 7(1): 48-56.

Edan, Y., D. Rogozin, T. Flash, and G. E. Miles. 2000. Robotic melon harvesting. *IEEE Trans. on Robotics and Automation* 16(6): 831-835.

Fallahi, E., B. Fallahi, and I.-J. Chun. 2006. Use of New Blossom Thinners to Reduce Fruit Set and Improve Fruit Quality in Apples, Peaches, Nectarines, and Plums. In *Intl. Symp. on Endogenous and Exogenous Plant Bioregulators*, 159-164. Seoul, Korea: ISHS.

Glozer, K., and J. Hasey. 2006. Mechanical thinning in cling peach. *HortScience* 41(4): 995-995.

Grift, T., Q. Zhang, N. Kondo, and K. Ting. 2008. A review of automation and robotics for the bioindustry. *J. of Biomechatronics Engineering* 1(1): 37-54.

Hamner, B., M. Bergerman, S. Singh, and G. House. 2011. Autonomous Orchard Vehicles for Specialty Crops Production. In *2011 ASABE Annual Intl. Meeting*. Louisville, KY: ASABE.

Hutchinson, S., G. D. Hager, and P. I. Corke. 1996. A tutorial on visual servo control. *IEEE Trans.on Robotics and Automation* 12(5): 651-670.

ISO. 1994. ISO 8373:1994 Manipulating industrial robots - Vocabulary. Geneva, Switzerland: ISO.

Kondo, N., M. Monta, and N. Noguchi. 2011. *Agricultural Robots: Mechanisms and Practices*. Melbourne, Australia: Trans Pacific Press.

Kondo, N., Y. Nishitsuji, P. Ling, and K. C. Ting. 1996. Visual feedback guided robotic cherry tomato harvesting. *Trans. ASAE* 39(6): 2331-2338.

Kubota, C., M. A. McClure, N. Kokalis-Burelle, M. G. Bausher, and E. N. Rosskopf. 2008. Vegetable grafting: History, use, and current technology status in North America. *HortScience* 43(6): 1664-1669.

Lely. 2011. Lely's 12,500th Astronaut Robot installed. Cambridgeshire, United Kindom: Lely. Available at: http://www.lely.com/en/home/media-centre/news-en-events. Accessed 24 June 2012.

Marini, R. P., and G. L. Reighard. 2008. Crop Load Management. In *The Peach: Botany, Production and Uses*, 289-302. D. R. Layne, and D. Bassi, eds. Cambridge, MA: CABI Publishing.

Monta, M., N. Kondo, and Y. Shibano. 1995. Agricultural robot in grape production system. In *1995 IEEE Intl. Conf. on Robotics and Automation*, 2504-2509 vol2503. Aichi, Japan: IEEE.

Nielsen, M., D. Slaughter, and C. Gliever. 2012. Vision-based 3D Peach Tree Reconstruction for Automated Blossom Thinning. *IEEE Trans.on Industrial Informatics* 8(1): 188-196.

ROBOTIS. 2006. USB2Dynamixel User's Manual. Ver. 1.2. Seoul, Korea.: ROBOTIS, Co., Ltd.

Schupp, J. R., T. A. Baugher, S. S. Miller, R. M. Harsh, and K. M. Lesser. 2008. Mechanical thinning of peach and apple trees reduces labor input and increases fruit size. *Horttechnology* 18(4): 660-670.

Sciavicco, L., and B. Siciliano. 2000. *Modelling and Control of Robot Manipulators*. Advanced textbooks in control and signal processing. Second ed. London, United Kindom: Springer.

Spong, M. W., S. Hutchinson, and M. Vidyasagar. 2006. *Robot modeling and control*. Hoboken, NJ: John Wiley & Sons.

Tung, C. P., and A. C. Kak. 1996. Integrating sensing, task planning, and execution for robotic assembly. *IEEE Trans. on Robotics and Automation* 12(2): 187-201.

USDA-NASS. 2012. Noncitrus Fruits and Nuts 2011 Preliminary Summary. ISSN: 1948-2698. Washington, D.C.: USDA National Agricultural Statistics Service.

Van Henten, E. J., J. Hemming, B. A. J. van Tuijl, J. G. Kornet, J. Meuleman, J. Bontsema, and E. A. van Os. 2002. An autonomous robot for harvesting cucumbers in greenhouses. *Autonomous Robots* 13(3): 241-258.

Van Henten, E. J., J. Hemming, B. A. J. Van Tuijl, J. G. Kornet, and J. Bontsema. 2003a. Collision-free motion planning for a cucumber picking robot. *Biosystems Eng.*86(2): 135-144.

Van Henten, E. J., B. A. J. Van Tuijl, J. Hemming, J. G. Kornet, J. Bontsema, and E. A. Van Os. 2003b. Field test of an autonomous cucumber picking robot. *Biosystems Eng.*86(3): 305-313.

APPENDIX A: Design of Peach Tree Model

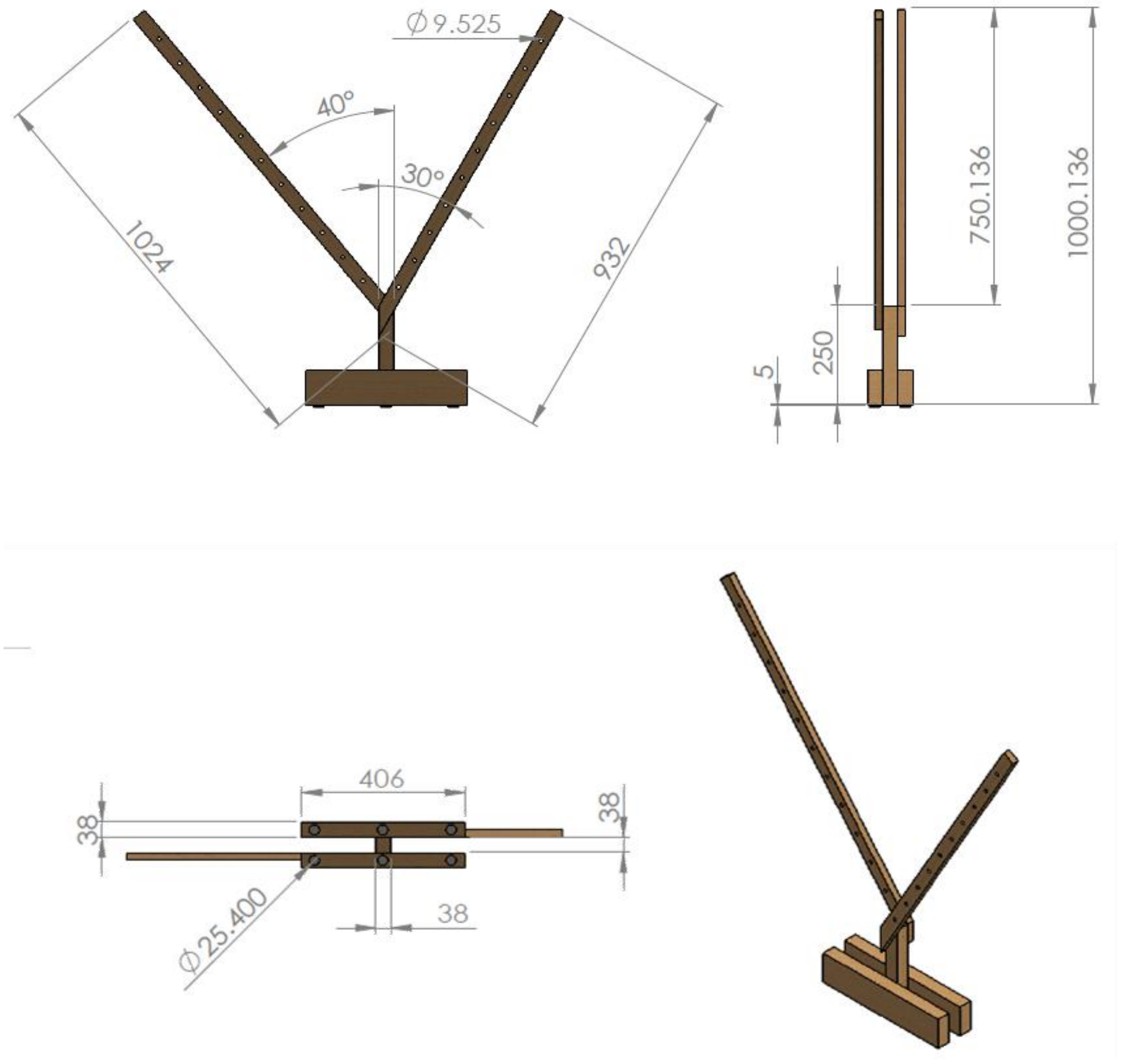


Figure A.1 Drawings of peach tree model. Dimensions are in mm.

APPENDIX B: Control Software

The execution procedures of the control software are summarized as a flowchart in Figure B.1. The program requires the manipulator's information, which includes the DH parameters and the transformation matrices being T_3^0 and T_6^0. The positions and orientations of target blossoms and branches are also required. The complete implementation is attached in the following pages.

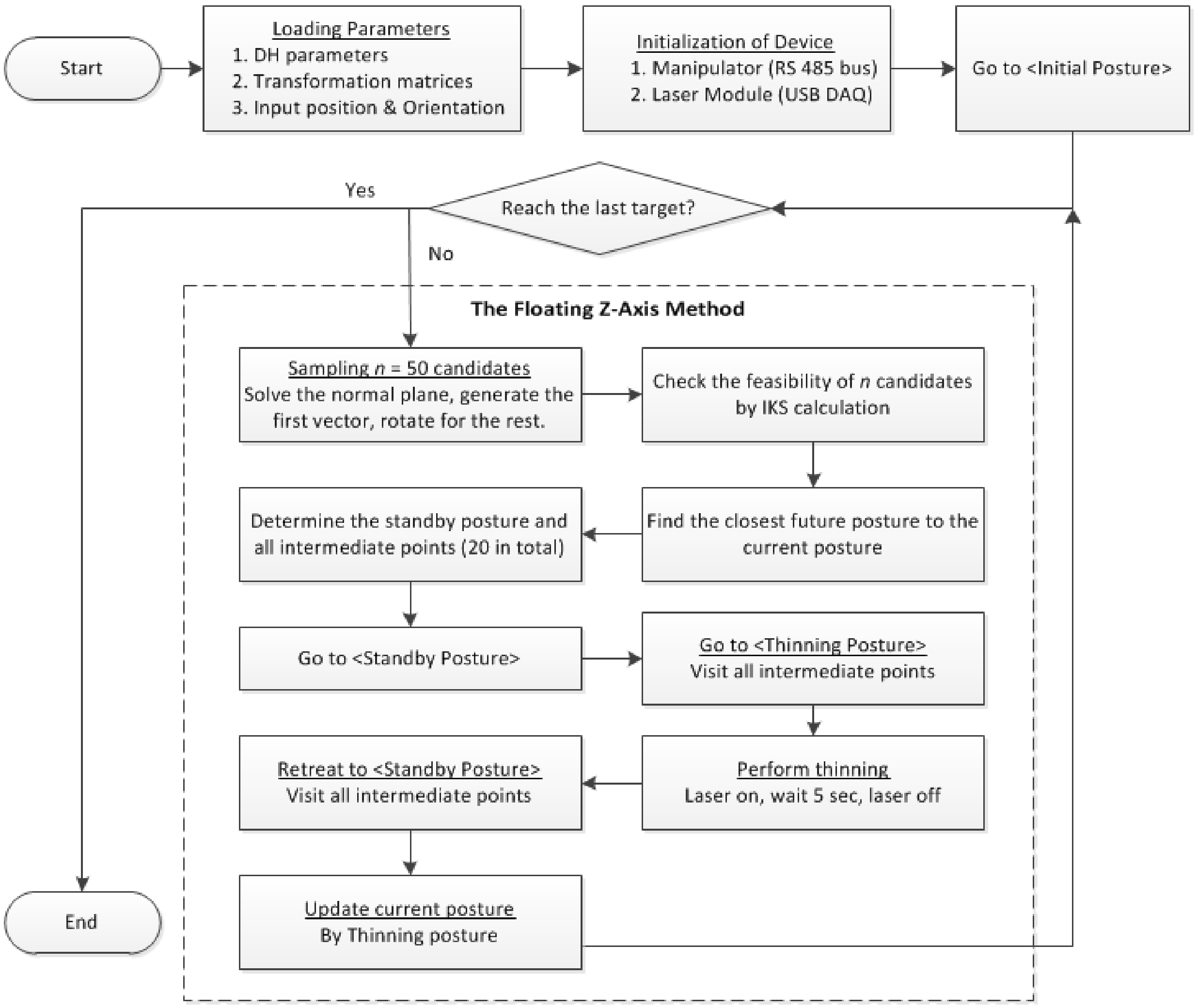

Start
Loading Parameters
1. DH parameters
2. Transformation matrices
3. Input position & Orientation
Initialization of Device
1. Manipulator (RS 485 bus)
2. Laser Module (USB DAQ)
Go to <Initial Posture>
Yes
Reach the last target?
No
The Floating Z-Axis Method
Sampling n = 50 candidates
Solve the normal plane, generate the first vector, rotate for the rest.
Check the feasibility of n candidates by IKS calculation
Find the closest future posture to the current posture
Determine the standby posture and all intermediate points (20 in total)
Go to <Standby Posture>
Go to <Thinning Posture>
Visit all intermediate points
Perform thinning
Laser on, wait 5 sec, laser off
Retreat to <Standby Posture>
Visit all intermediate points
Update current posture
By Thinning posture
End

Figure B.1 Flowchart of the control software.

```
function MainControl_v09
% Filename: MainControl_v09
% Programmer: Fu Ouyang
% Date: June 24, 2012
% Description: This version is a standalone function comprise all the
%              other subroutines (also functions )in the same script.
%              Beyond this, it is identical to MainControl_v08

%% Loading & Setting
% --- Adding Directory ---
% addpath('Subroutine_Customized'); % Not necessaty for this version

% --- Loading DH parameters (Generated by FK_v04.m) ---
addpath('Load');
load('DH_Parameters'); % It includes: a, af, d
d(6) = d(6) + 4;    % length of end-effector

% --- Loading Transformation Matrix ((Generated by FK_v04.m)) ---
addpath('Load');
load('TransMat');
syms th1 th2 th3 th4 th5 th6; % variables in T03, T06

% --- Limit of Joint Variables (q's) in degree ---
LimU = [ 135; 100;  90;  150;  91;  150]';
LimL = [-135; -45; -99; -150; -91; -150]';

% === User Input ===
% An complete input includes a 3 by 3 orientation matrix R06 and a 3 by 1
% position vector of the endeffector ptEF as follows.
%                              |- rx1  ry1  rz1 -|
% R06 =  [  Rx   Ry   Rz   ] = |  rx2  ry2  rz2  |
%                              |- rx3  ry3  rz3 -|
% ptEF = [px; py; pz]
%  --- Data set 3 ---
inPtEF(:, 1) = [18.5;  -5.7;  42.0];
inOrt(:,1)   = [  -1;   -1;  1];

inPtEF(:, 2) = [18.0;  -7.0;  42.5];
inOrt(:,2)   = [  -1;   -1;  1];

inPtEF(:, 3) = [16.0;  -8.3; 43.5];
inOrt(:,3)   = [  -1;   -1;  0];

inPtEF(:, 4) = [15.0;  -8.9; 44.5];
inOrt(:,4)   = [  -1;   -1;  0];

inPtEF(:, 5) = [14.0;  -9.5; 45.0];
inOrt(:,5)   = [  -1;   -1;  0];

%% Initialization of Device
% === Initializing Robot ===
robotStatus('on');  % subroutine

% --- Parameter Setting (for Motor)
% Address value (30)saves the goal position of Dynamixel's Control Table.
P_GOAL_POSITION = 30;
P_Moving = 46;

% --- Define Speed level ---
DefaultSpd = zeros(6,5);
DefaultSpd(:,1) = [5,5,5,5,5,5]';
DefaultSpd(:,2) = [10,10,10,10,10,10]';
DefaultSpd(:,3) = [25,25,25,25,25,25]';
DefaultSpd(:,4) = [50,50,50,50,50,50]';
DefaultSpd(:,5) = [70,70,70,70,70,70]';

% --- Calibration Offset ---
% The values here were determined by measuring the errors from rulers.
CalOff = zeros(6,1);
CalOff(1) = 1;  CalOff(2) = 0;  CalOff(3) = 0;
CalOff(4) = 0;  CalOff(5) = 0;  CalOff(6) = 0;

% === Initializing End-effector ===
% NI USB-6008 pin of "ao1" & "GND"
ao = analogoutput('nidaq', 'Dev4');
addchannel(ao,1);
set(ao, 'TriggerType', 'Immediate');
putsample(ao,5);



%% Path Planning and Moving
%  === Initial Posture ===
R06       = [-1 0 0; 0 -1 0; 0 0 1]';
ptEF      = [ 11.60;      0; 39.50];
JV        = nan(1, 6);

% --- Move the manipulator ---
[JVcandidates, ~]  = iksdecision(R06, ptEF, a, af,d, T03, LimU, LimL);
[~, ind] = min(sum(abs(JVcandidates),2));
JV(1,:) = JVcandidates(ind,:);
InputPos = angleConv( JV(1,:) ) + CalOff;
SeqInstr = [ InputPos DefaultSpd(:,2)];

commander(SeqInstr,P_GOAL_POSITION, P_Moving, ao) % instruct the robot

for location = 1 : size(inPtEF,2)
    %  === Intermediate Points ===
    numRdn      = 50;
    [ortMat, ~] = ortgen_v2(inPtEF(:,location), inOrt(:,location), numRdn);
    inPtEFmat   = repmat(inPtEF(:,location), 1, numRdn);
    tic
```

```
[JVcandidates, JVsrc] = iksdecision(ortMat, inPtEFmat, a, af,d, ...
                        T03, LimU, LimL);
toc

% --- Find the closest configuration to the current one ---
dist        = cdist(JVcandidates, JV(1,:));     % custom subroutine
[~, ind]    = min(dist);
JV(2,:)     = JVcandidates(ind,:);



% --- Find the Standy Posture and all intermediate points ---
standbyDist = 4;
numStop     = 20;
deltaStep   = standbyDist/numStop;
for stop = 1:numStop
    clearance = standbyDist - deltaStep*(stop-1);
    zVec      = ortMat(:, 3, JVsrc(ind));
    t         = [clearance/norm(zVec), -clearance/norm(zVec)];
    pt        = [zVec*t(1) + inPtEF(:,location), ...
                 zVec*t(2) + inPtEF(:,location)];
    [~, idx]  = min( [norm(pt(:,1)), norm(pt(:,2))] );

    [JVtemp,~]= iksdecision(ortMat(:,:,JVsrc(ind)), pt(:,idx), ...
                            a, af,d, T03, LimU, LimL);
    dist      = cdist(JVtemp, JV(1+stop,:));
    [~, id]   = min(dist);
    JV(2+stop,:)= JVtemp(id,:);
end

InputPos = nan(6,numStop*2+1);
SeqInstr = nan(6,2,numStop*2+1);
for pos = 1:numStop
    InputPos(:,pos)         = angleConv( JV(2+pos,:) );
    SeqInstr(:,:,pos)       = [ InputPos(:,pos) DefaultSpd(:,1)];
```

```
        InputPos(:,end+1-pos)   = angleConv( JV(2+pos,:) );
        SeqInstr(:,:,end+1-pos) = [ InputPos(:,pos) DefaultSpd(:,1)];
    end
    InputPos(:,numStop+1)   = angleConv( JV(2,:) );
    SeqInstr(:,:,numStop+1) = [ InputPos(:,numStop+1) DefaultSpd(:,1)];

    commander(SeqInstr,P_GOAL_POSITION, P_Moving, ao)
    JVcurrent      = JV(3,:);
    JV             = nan(1, 6);
    JV(1,:)        = JVcurrent;
end

%% Terminating Device
robotStatus('off');

% =========== End of Main ==========


%% Switch on/off the manipulator
function robotStatus(str)
switch str
    case (Lely, 2011Lely, 2011)   % Turn on the robot
        % --- Import the SDK (locate the "dynamixel.dll") ---
        addpath('import')
        addpath('Subroutine_Customized')

        % --- Load Library ---
        loadlibrary('dynamixel', 'dynamixel.h');

        % --- Parameter Setting
        DEFAULT_PORTNUM = 3;        % COM3
        DEFAULT_BAUDNUM = 1;        % BUS speed: 1 Mbps
        % --- Open Device ---
        calllib('dynamixel','dxl_initialize', ...
                DEFAULT_PORTNUM,DEFAULT_BAUDNUM);

    case {'off', 'Off'} % Turn off the robot
        % Close Device
        calllib('dynamixel','dxl_terminate');

        % Unload Library when the program is ended.
        unloadlibrary('dynamixel');

    otherwise
        error('Input argument can only be string "on" or "off."');
end


%% Inverse Kinematics Solution Selector
function [Q_degDepot, Q_srcDepot] = iksdecision(R06, ptEF, a, af,d,...
                                                 T03, LimU, LimL)
numLocation = size(R06, 3);

Q_degDepot  = [];
Q_srcDepot  = [];
for location = 1 : numLocation
    [Q_deg, ~]= invKinematics(R06(:,:,location), ptEF(:,location),...
                              a, af,d, T03);

    for solution = 1:4
        % Within limit check
        CheckU = Q_deg(solution,:) <= LimU;
        CheckL = Q_deg(solution,:) >= LimL;
        if sum(CheckU + CheckL) == 12
            Q_degDepot = [Q_degDepot; Q_deg(solution,:)];
            Q_srcDepot = [Q_srcDepot; location];
        end
    end
end

%% Inverse Kinematics Solution Calculator
function [Q_deg, Q_flag] = invKinematics(R06, ptEF, a, af, d, T03)
% === Documentation ===
% --- Reference ---
% Spong, M., S. Hutchinson, M. Vidyasagar.
% Robot Modeling and Control. 2006. Wiley.

% === Declaration ===
a1 = a(1);  %  1.35
d1 = d(1);  % 11.40
a2 = a(2);  % 17.75
d4 = d(4);  % 10.25; a3 and d4 are exchangeble
d6 = d(6);  %  6.40

syms th1 th2 th3 th4 th5 th6; % variables in T03

% === IKS Calculation ===
ptWC = ptEF - d6 * R06(:,3);       % location of Wrist Center
xc = ptWC(1);
yc = ptWC(2);
zc = ptWC(3);

% --- Prelocating Solution Matrix ---
% There are only three conditions for solving IK; they are 4 solution sets,
% 2 sets, or infinite sets.
% <Note 1>: q3 are dependent to q2, which usually has two solutions.
% <Note 2>: q4 & q6 are dependent to q5, which usually has two solutions.
Q = zeros(4, 6);
    %               q1   q2    q3    q4    q5    q6
    % solution 1    q1   q2_1  q3_1  q4_1  q5_1  q6_1
    % solution 2    q1   q2_1  q3_1  q4_1  q5_2  q6_1
    % solution 3    q1   q2_2  q3_2  q4_2  q5_3  q6_2
    % solution 4    q1   q2_2  q3_2  q4_2  q5_4  q6_2
Q_flag = ones(4,1);  % Indicates whether solution is useful or redundant.

% --- q1 ---
q1 = atan2(yc, xc);
Q(:,1) = q1;

% --- q3 ---
r = sqrt(xc^2 + yc^2) - a1;
s = zc - d1;
D = (r^2 + s^2 -a2^2 -d4^2)/(2*a2*d4);
q3(1) = atan2(+sqrt(1-D.^2), D);
q3(2) = atan2(-sqrt(1-D.^2), D);
Q([ 9 10]) = q3(1);
Q([11 12]) = q3(2);

% --- q2 ---
q2(1) = atan2(s,r) - atan2(d4*sin(q3(1)), a2+d4*cos(q3(1)));
q2(2) = atan2(s,r) - atan2(d4*sin(q3(2)), a2+d4*cos(q3(2)));
Q([5 6]) = q2(1);
Q([7 8]) = q2(2);

% --- q5(1), q5(2) ---
% Reference: p54-p56, p107-p108
T03val_s1 = subs(T03, {th1, th2, th3}, {q1, q2(1), q3(1)+ deg2rad(90)});
R03_s1 = T03val_s1(1:3,1:3);
R36_s1 = R03_s1'* R06;
R =  R36_s1;


r11 = R(1,1);   r12 = R(1,2);   r13 = R(1,3);
r21 = R(2,1);   r22 = R(2,2);   r23 = R(2,3);
```

```
r31 = R(3,1);   r32 = R(3,2);   r33 = R(3,3);

if abs( r13 + r23 ) < 10^-10  % Both R(1,3) and R(2,3) is 0
    q5(1) = 0;
    q5(2) = 0;
    Q_flag(2) = 0;    % Set solution 2 as redundant.
    if abs( r33 - 1 ) < 10^-10   % R(3,3) is 1
        % --- q4 + q6 = Atan2(-r12,r11) ---
        q46 = atan2(-r12, r11);
        q4(1) = 0;
        q6(1) = q46 - q4(1);
        q4(2) = q4(1);
        q6(2) = q4(1);
    else                         % R(3,3) is -1
        % --- q4 - q6 = Atan2(-r12,-r11) ---
        q46 = atan2(-r12, -r11);
        q4(1) = pi;
        q6(1) = q4(2) - a46;
        q4(2) = q4(1);
        q6(2) = q6(1);
    end
else
    % --- Solution Set 1 ---
    q5(1) = atan2( + sqrt( 1 - (r33)^2 ), r33 );
    q4(1) = atan2( r23, r13 );
    q6(1) = atan2( r32, -r31);
    % --- Solution Set 2 ---
    q5(2) = atan2( - sqrt( 1 - (r33)^2 ), r33 );
    q4(2) = atan2( -r23, -r13);
    q6(2) = atan2( -r32, r31);
end

Q(13) = q4(1);    Q(17) = q5(1);    Q(21) = q6(1);
Q(14) = q4(2);    Q(18) = q5(2);    Q(22) = q6(2);
```

```
% --- q5(3), q5(4) ---
T03val_s2 = subs(T03, {th1, th2, th3}, {q1, q2(2), q3(2) + deg2rad(90)});
R03_s2 = T03val_s2(1:3,1:3);
R36_s2 = R03_s2'* R06;
R = R36_s2;

r11 = R(1,1);   r12 = R(1,2);   r13 = R(1,3);
r21 = R(2,1);   r22 = R(2,2);   r23 = R(2,3);
r31 = R(3,1);   r32 = R(3,2);   r33 = R(3,3);

if abs( r13 + r23 ) < 10^-10  % Both R(1,3) and R(2,3) is 0
    q5(3) = 0;
    q5(4) = 0;
    Q_flag(4) = 0;    % Set solution 4 as redundant.
    if abs( r33 - 1 ) < 10^-10   % R(3,3) is 1
        % --- q4 + q6 = Atan2(-r12,r11) ---
        q46 = atan2(-r12, r11);
        q4(3) = 0;
        q6(3) = q46 - q4(3);
        q4(4) = q4(3);
        q6(4) = q6(3);
    else                        % R(3,3) is -1
        % --- q4 - q6 = Atan2(-r12,-r11) ---
        q46 = atan2(-r12, -r11);
        q4(4) = pi;
        q6(4) = q4(4) - q46;
        q4(4) = q4(3);
        q6(4) = q6(3);
    end
else
    % --- Solution Set 3 ---
    q5(3) = atan2( + sqrt( 1 - (r33)^2 ), r33 );
    q4(3) = atan2( r23, r13 );
    q6(3) = atan2( r32, -r31);
    % --- Solution Set 4 ---
    q5(4) = atan2( - sqrt( 1 - (r33)^2 ), r33 );
    q4(4) = atan2( -r23, -r13);
    q6(4) = atan2( -r32, r31);
end

Q(15) = q4(3);    Q(19) = q5(3);    Q(23) = q6(3);
Q(16) = q4(4);    Q(20) = q5(4);    Q(24) = q6(4);

Q_deg = rad2deg(Q);


%% Joint/Motor angle convertor
function thetaOut = angleConv(thetaIn)
% A motor has its middle angle at 12-o-clock position while a joint has
% zero degree there.

thetaOut = zeros(6,1);

thetaOut(1) =  thetaIn(1)+150;
thetaOut(2) =  thetaIn(2)+125.46;
thetaOut(3) = -thetaIn(3)+60;
thetaOut(4) = thetaIn(4)+150;
thetaOut(5) =  thetaIn(5)+150;
thetaOut(6) =  thetaIn(6)+150;


%% Actuation of Manipulator
function commander(SeqInstr,P_GOAL_POSITION, P_Moving,ao)
% === Feeding Command ===
numStep = size(SeqInstr,3);

for instr = 1 : numStep
```

```
tic

% --- Instruct individual motor ---
[ id, GoalPos, MovingSpd ] = joint2motor( SeqInstr(:,:,instr) );

% --- Make syncwrite packet ---
Qty = size(id,2);
BROADCAST_ID = 254;   % Not sure about the meaning of this value.
calllib('dynamixel', 'dxl_set_txpacket_id', BROADCAST_ID);
instrLength = 4;
calllib('dynamixel','dxl_set_txpacket_length',(instrLength + 1)*Qty+4);
INST_SYNC_WRITE = 131;
% Write command into motor register
calllib('dynamixel', 'dxl_set_txpacket_instruction', INST_SYNC_WRITE);
% Starting address to write Data
calllib('dynamixel', 'dxl_set_txpacket_parameter', 0, P_GOAL_POSITION);
% Length of data for each motor
calllib('dynamixel', 'dxl_set_txpacket_parameter', 1, instrLength);

for i = 0 : Qty-1  % This loop build the data for each motor.
    % Write ID
    calllib('dynamixel', 'dxl_set_txpacket_parameter',...
            2+(instrLength+1)*i, id(i+1));

    % Write Goal Positition
    low = calllib('dynamixel', 'dxl_get_lowbyte', GoalPos(i+1));
    calllib('dynamixel', 'dxl_set_txpacket_parameter', ...
            2+(instrLength+1)*i+1, low);
    high = calllib('dynamixel', 'dxl_get_highbyte', GoalPos(i+1));
    calllib('dynamixel', 'dxl_set_txpacket_parameter', ...
            2+(instrLength+1)*i+2, high);

    % Write Moving Speed
    low = calllib('dynamixel','dxl_get_lowbyte', MovingSpd(i+1));
```

```
        calllib('dynamixel', 'dxl_set_txpacket_parameter', ...
                2+(instrLength+1)*i+3, low);
        high = calllib('dynamixel','dxl_get_highbyte', MovingSpd(i+1));
        calllib('dynamixel', 'dxl_set_txpacket_parameter', ...
                2+(instrLength+1)*i+4, high);
    end

    calllib('dynamixel', 'dxl_txrx_packet');
    t1(instr) = toc;        % Recording the time of encoding.

    % Check moving done
    movFlag = 8;            % A flag indicates how many motors are moving.
    while (movFlag ~=0)
        movFlag = 0;
        for i = 1:8
            movFlag = movFlag + ...
                      int32(calllib('dynamixel','dxl_read_byte',...
                            id(i),P_Moving));
        end
    end

    t2(instr) = toc;        % Recording the time of excution.

    % Thinning Laser on
    if instr == ceil(numStep/2)
        putsample(ao,0);
        pause(5)
        putsample(ao,5);
    end
end


%% Joint/Motor Degree conversion
function [ id, GoalPos, MovingSpd] = joint2motor( CommandTable )
```

```
% Task:
% (1) Building instruction matrix for feeding into communication packet.
% (2) Conversion of: Moving Speed from RPM to Dynamixel motor scale.
% (3) Conversion of: Goal Position from degree to  Dynamixel motor scale.
% Be careful:
% The input data is the information about "Joint" rather than motor. Since
% there are some joints have two motors, the dimension of output data will
% be larger than the input.

% --- Initialize Outputs ---
GoalPos = [];
MovingSpd = [];
id = [];

for i = 1:6
% Inside this loop, two subroutines are called: PosConvert & SpdConvert
    switch i
        case (1)    % Joint 1: RX-64
            Pos11 = PosConvert('RX-64', CommandTable(i,1));
            Spd11 = SpdConvert('RX-64', CommandTable(i,2));
            GoalPos = [GoalPos, Pos11];
            MovingSpd = [MovingSpd, Spd11];
            id = [id, 11];
        case (2)    % Joint 2: CAUTION! two back to back EX-106+
            Pos21 = PosConvert('EX-106+', CommandTable(i,1));
            Pos22 = abs(4095-Pos21);        % Be careful about "4095"!
            Spd21 = SpdConvert('EX-106+', CommandTable(i,2));
            Spd22 = SpdConvert('EX-106+', CommandTable(i,2));
            GoalPos = [GoalPos, Pos21, Pos22];
            MovingSpd = [MovingSpd, Spd21, Spd22];
            id = [id, 21, 22];
        case (3)    % Joint 3: CAUTION! two back to back RX-64
            Pos31 = PosConvert('RX-64', CommandTable(i,1));
            Pos32 = abs(1023-Pos31);        % Be careful about "1023"!
            Spd31 = SpdConvert('RX-64', CommandTable(i,2));
            Spd32 = SpdConvert('RX-64', CommandTable(i,2));
            GoalPos = [GoalPos, Pos31, Pos32];
            MovingSpd = [MovingSpd, Spd31, Spd32];
            id = [id, 31, 32];
        case (4)     % Joint 4: RX-64
            Pos41 = PosConvert('RX-64', CommandTable(i,1));
            Spd41 = SpdConvert('RX-64', CommandTable(i,2));
            GoalPos = [GoalPos, Pos41];
            MovingSpd = [MovingSpd, Spd41];
            id = [id, 41];
        case (5)     % Joint 5: RX-64
            Pos51 = PosConvert('RX-64', CommandTable(i,1));
            Spd51 = SpdConvert('RX-64', CommandTable(i,2));
            GoalPos = [GoalPos, Pos51];
            MovingSpd = [MovingSpd, Spd51];
            id = [id, 51];
        case (6)     % Joint 6: RX-28
            Pos61 = PosConvert('RX-28', CommandTable(i,1));
            Spd61 = SpdConvert('RX-28', CommandTable(i,2));
            GoalPos = [GoalPos, Pos61];
            MovingSpd = [MovingSpd, Spd61];
            id = [id, 61];
    end
end

function Pos_dynamixel = PosConvert(motorType, Pos_deg)
switch motorType
    case {'RX-28', 'RX-64'}
        Pos_dynamixel = 1023/300*Pos_deg;
    case {'EX-106+'}
        Pos_dynamixel = 4095/250.92*Pos_deg;
end
```

```
function Spd_dynamixel = SpdConvert(motorType, Spd_rpm)
    Spd_dynamixel = Spd_rpm/0.111;



%% Scouting the Normal Plane
function [ortMat, vecMat] = ortgen_v2(inPt, inOrt, numOrt)
%% Variables Initialization
% ux + vy + wz = k; representation of normal plane
% k = inOrt(1)*inPt(1) + inOrt(2)*inPt(2) + inOrt(3)*inPt(3)
k = inOrt' * inPt;

% == Find the Normal Plane ==
checkZero = find(inOrt == 0);

vecMat = zeros(3, numOrt);
rPtMat = zeros(3, numOrt);
ortMat = zeros(3,3, numOrt);

% === Decide the first vector ===
% --- Generate a random starting point ---
if length(checkZero) == 2
    if inOrt(1) ~= 0;       % plane: ux = k
        px = inPt(1);
        py = randx(1);
        pz = randz(1);
    elseif inOrt(2) ~= 0;   % plane: vy = k
        py = inPt(2);
        px = randx(1);
        pz = randz(1);
    else                    % plane: wz = k
        pz = inPt(3);
        py = randy(1);
        px = randx(1);
    end
```

```
elseif length(checkZero) == 1
    if inOrt(1) == 0;        % plane: vy + wz = k
        px = randx(1);
        py = randy(1);
        pz = (1/inOrt(3)) * (k - inOrt(2)*py);
    elseif inOrt(2) == 0;    % plane: ux + wz = k
        py = randy(1);
        px = randx(1);
        pz = (1/inOrt(3)) * (k - inOrt(1)*px);
    else                     % plane: ux + vy = k
        pz = randz(1);
        py = randy(1);
        px = (1/inOrt(1)) * (k - inOrt(2)*py);
    end
else                         % plane: ux + vy + wz = k
    px = randx(1);
    py = randy(1);
    pz = (1/inOrt(3)) * (k - inOrt(1)*px - inOrt(2)*py);
end

% --- Generate the first vector ---
randPt       = [px; py; pz];
vec          = randPt - inPt;
vec          = vec/norm(vec); % Potential orientation for the end-effector

% === Decide the first orientation of the tool frame===
x_comp        = cross(inOrt, vec);
ortTemp       = [x_comp/norm(x_comp), inOrt/norm(inOrt), vec];
ortMat(:,:,1) = ortTemp;

% --- Generate the rest numOrt-1 vectors ---
deg = 360/numOrt;
% Rotatiion along y-axis (plane normal vector)
rot = [ cosd(deg)   0  sind(deg); ...
```

```
                0   1          0; ...
       -sind(deg)   0  cosd(deg)];

for iter = 2 : numOrt
    ortTemp          = ortTemp * rot;
    ortMat(:,:,iter) = ortTemp;
end

vecMat = squeeze(ortMat(:,3,:));
disp(vecMat);

function px = randx(numPx)
    maxX = 50;
    px = maxX * rand(numPx);

function py = randy(numPy)
    maxY = 50;
    py = (2*maxY)*rand(numPy) - maxY;

function pz = randz(numPz)
    maxZ = 50;
    pz = (2*maxZ)*rand(numPz) - maxZ;


%% C-space Distance Calculator
function dist = cdist(JVcandidates, JVref)
% JVcandidates is an m-by-6 matrix (m can be 1 well)
% JVref        is a  1-by-6 matrix
% dist         is a  m-by-1 vector
JVrefMat = repmat(JVref, size(JVcandidates, 1),1);
dist     = sum(abs(JVcandidates - JVrefMat), 2);

% ========== End ==========
```

APPENDIX C: Personnel

This appendix provides contact information of project participants outside UIUC. Dr. Paul Heinemann is the principal investigator of the ITTF project; Dr. James Schupp provided support with horticultural knowledge and experience; David Lyons is a doctoral student at PSU who is working on the development of the end-effector.

Paul Heinemann, Ph.D.
Professor and Department Head
Department of Agricultural and Biological Engineering
Pennsylvania State University
http://abe.psu.edu/directory/hzh
hzh@psu.edu

James Schupp, Ph.D.
Associate Professor of Pomology
Fruit Research and Extension Center
Pennsylvania State University
http://horticulture.psu.edu/directory/jrs42
jrs42@psu.edu

David J. Lyons
Ph.D candidate
Department of Agricultural and Biological Engineering
Pennsylvania State University
djl272@psu.edu