2950 Niles Road, St. Joseph, MI 49085-9659, USA
269.429.0300 fax 269.429.3852 hq@asabe.org www.asabe.org

An ASABE Meeting Presentation
DOI: https://doi.org/10.13031/aim.202000439
Paper Number: 2000439

Collision free Path Planning of a Robotic Manipulator for Pruning Apple Trees

Azlan Zahid [1,3], Long He [1,3], Daeun Dana Choi [1], James Schupp [2,3], Paul Heinemann [1]
[1] Department of Agricultural and Biological Engineering, Pennsylvania State University, University Park, PA 16802, USA
[2] Department of Plant Science, Pennsylvania State University, University Park, PA 16802, USA
[3] Penn State Fruit Research and Extension Center (FREC), Biglerville, PA 17307, USA
**** Correspondence: axz264@psu.edu***

Written for presentation at the
2020 ASABE Annual International Meeting
Sponsored by ASABE
Omaha, Nebraska
July 12–15, 2020

ABSTRACT. *Pruning of apple trees requires 80-120 working hours of labor per hectare accounting for 20% of the total production cost. Robotic pruning is a potential solution to decrease labor dependence and associated costs. Autonomous precise manipulation of a robotic manipulator in presence of obstacles is a challenge. The spatial requirements and collision-free path planning for the robotic manipulator is essential for automated systems. This simulation study focused on investigating the branch accessibility of a six-rotational (6R) degrees of freedom (DoF) robotic manipulator with a shear blade type end-effector. A virtual tree canopy environment was established in MATLAB for simulation. The Rapidly-exploring Random Tree (RRT) obstacle avoidance algorithm was used to establish a collision-free path to reach the target pruning points. The path smoothing and optimization algorithms were also used to reduce path length and calculate the optimize path. The simulation showed that the integrated robotic manipulator reached the pruning points avoiding obstacle untargeted branches. The path generation time, path length, target reaching time, and number of accessible branches (success) and collisions (failure) was recorded. The study provides the foundation information for future work on the development of a robotic pruning system*

Keywords. *Branch accessibility, Manipulator path planning, Robotic pruning, Virtual tree environment.*

1. Introduction

In the year 2019, the apple industry contributed approximately $3.01 billion to the United States' economy (USDA-NASS, 2019). The production operations including pruning, thinning and harvesting of most tree fruits still largely depend on manual labor (Silwal, 2016). Manual pruning of fruit trees requires 80-120 working hours of skilled labor per hectare (Mika et al., 2016). Tree fruit growers are facing serious challenges with limited labor pool and associated costs to the sustainability of the tree fruit industry. Therefore, alternative solutions for pruning task is essential. Robotic pruning is a selective branch pruning operation to cut the branches using a shear blade end-effector (Lehnert, 2012). The end-effector is

integrated with a manipulator which can move the end-effector within the canopy to reach a target location following a safe path, and a vison system for 3D canopy reconstruction is also included. The key physical components of a robotic pruning machine are manipulators (robotic arm) and the end-effector (tool). Robotic systems are now extensively studied in agriculture, however, their success is limited due to challenges such as natural variability in tree structure and illumination (Bac et al., 2014; Kapach et al., 2012; Li et al., 2011).

Previous studies have reported on 3D reconstruction of branches for pruning with machine vision systems (Karkee et al., 2014; Lindner et al., 2007; Nakarmi and Tang, 2012). Only a few studies focused on using robotic manipulators for tree pruning (Bac et al., 2014; He and Schupp, 2018; Kondo et al., 1993, 1994). Most of these studies focused on pruning grape vines (Kondo et al., 1993; Vision Robotics Corporation, 2015). These systems worked for pre-defined targets, and a point-to-point path method was used for controlling the trajectory of the robotic manipulator and collision avoidance was unnecessary. In contrast, the adoption of a robotic manipulator for apple tree pruning is challenging mainly due to the complex canopy working environment and spatial requirements of the manipulator (He and Schupp, 2018). The tree structure is complex and branches usually crisscross, causing the manipulator and end-effector to collide with branches easily. The spatial requirements for maneuvering of the manipulator and the end-effector should be considered to avoid collisions with branches. The collision may damage the branches or greatly reduce manipulator performance, thus affecting the quality of pruning operation (Gongal et al., 2016). Therefore, a collision-free path should be planned for the manipulator to avoid collisions with untargeted branches to reach the target branches at specified locations (Cao et al., 2019).

The collision-free path planning for robotic pruning refers to the movement of the integrated manipulator and end-effector from a known starting position to a known target position following a trajectory without collision. In other words, there is no branch intersection to the end-effector tool-center-point (TCP) path, and no side collision to the robot body with the branches. By satisfying these two requirements, the robot can move from a home position to the target pruning location in a collision free manner with optimal or near optimal cost. The optimal criteria for robots depend on the physical constraints and applications of the robot. The criteria could consist of one or more conditions including physical distance, smoothness, risk, and fuel requirements (Noreen et al., 2016b). Hence, path planning for robots refers to find a feasible path according to application specified criterion under the influence of the holonomic constraints, such as robotic manipulators or non-holonomic constraints such as mobile robots (LaValle, 2006). Autonomous precise manipulation in the presence of obstacles is a great challenge. Path planning algorithms are widely used in the automation and artificial intelligence industry for motion planning of numerous system such as autonomous cars, UAVs, planetary and space missions (Noreen et al., 2016). The early path planning methods such as Cell Decomposition (CD), Potential Fields (PF), and Road Map (RM) algorithm did not perform well with dynamic and high dimension applications due to computational complexity and thus are limited only to low dimensional problems (Elbanhawi and Simic, 2014; Goerzen et al., 2009; Karaman and Frazzoli, 2011; Kuffner and La Valle, 2000).

In the last few decades, some efforts have been put on improving the path planning algorithms, including grid-based and random sampling algorithms for dynamically complex environments (Nasir et al., 2013). Van Henten et al. (2003) used grid-based A* method for path planning of a cucumber-picking manipulator in a greenhouse. The method provided satisfactory results only for low DoF (2-3 DoF) manipulators as the computational complexity increases exponentially with the increasing of DoF (Cao et al., 2019). The path planning based on the random sampling approaches exhibits higher success rates for high dimensional problems, and has low computational cost (Elbanhawi and Simic, 2014; Karaman and Frazzoli, 2011). Among sample-based planning approaches, probabilistic roadmap, rapidly exploring random tree (RRT), and RRT star (RRT*) have been widely adopted for path planning. The probabilistic roadmap approach supports static and structured environments whereas RRT and RRT* methods supports unstructured and dynamic environments (Noreen et al., 2016).

LaValle (1998) proposed an RRT algorithm for path planning and it showed high efficiency compared to other sample-based methods (Cao et al., 2019; LaValle, 1998; Noreen et al., 2016b).Yang et al. (2017) implemented the RRT algorithm for path planning of a hybrid serial-parallel harvesting manipulator and realized the effectiveness of RRT path planning in high dimensional dynamic problems. Cao et al. (2019) also used an RRT algorithm along with a genetic algorithm for path planning of a robotic manipulator for litchi harvesting. At present, the RRT algorithm has been widely used in the field of robot path planning (Wu et al., 2016). However, thse path solutions of the RRT algorithm are not always smooth and optimal. The tree exploration is based on random searching of points in a configuration space which results in more computational time and low convergence speed. Considering the deficiencies of the RRT algorithm, improved solutions have been reported by combining RRT with several smoothing and optimization methods (Carbone et al., 2008; Choi et al., 2000; Gómez-Bravo et al., 2012; Karaman and Frazzoli, 2011; Saramago and Ceccarelli, 2004). Cao et al. (2019) presented an RRT smoothing method based on removing the unnecessary nodes to reduce the path length. A cubic polynomial trajectory procedure proposed to reduce the travelling time in a physically constrained environment (Lin et al., 1983). The path smoothing and optimization methods were succesfully implemented for path planning of multi-DoF manipulators. However, only a few studies have been reported for the adoption of the RRT path planning in an unstructured environment such as agriculture, and no study has been reported for RRT path planning for pruning trees. An orchard is an unstructured and dynamic environment, so a collision free trajectory for multi-DoF robot could be possibly established using the RRT algorithm.

The primary goal of the study was to establish a collision-free trajectory for a robotic pruning manipulator to move from

starting point to a target pruning position with presence of obstacles. The procedure is based on combining the RRT algorithm with a smoothing and optimization method, which aims to reduce the computational time and path length. The objectives of the study include: 1) Developing a simplified virtual environment including a robotic manipulator and a tree section for simulation in MATLAB; 2) Establishing a collision-free trajectory for reaching the targeted pruning points. The paper is organized as follows. In Sections 2, the manipulator model and obstacle environment are established, Section 3, the RRT planning method combined with smoothing and optimization is described. Section 4 presents the simulation process in a virtual environment. Section 5 illustrates the results and discussion of the simulation. The paper closes with conclusions and future directions.

2. Simulation Model Establishment

2.1 Robotic Manipulator

A six rotational (6R) DoF industrial robotic manipulator (UR-5, Universal Robots, Odense, Denmark) was used in this study. The manipulator has 6R joints where joint 1 is the base of the robotic manipulator and a shear pruner end-effector is coupled to joint 6 to reach the target pruning points. The three-dimensional model of the manipulator is shown in Figure 1 (a). The cutter end-effector is aligned parallel to the YZ_T plane of the tool frame (X_T, Y_T and Z_T). As shown in Figure 1(b), the outer dimension of the cutter is 200 x 50 x 50 mm (L1 x L2 x L3).

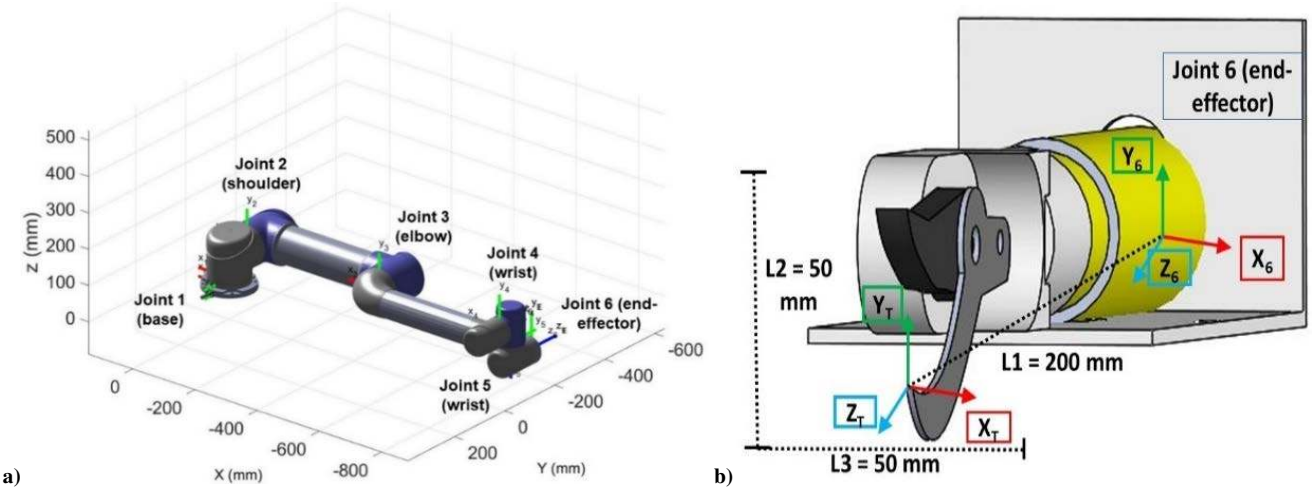

Figure 1. Three-dimensional model. (a) UR5 manipulator, and (b) End-effector

The frame representation of the manipulator kinematic model is shown in Figure 2. The Denavit-Hartenberg (DH) parameters of the UR5 manipulator calculated using the method presented by Craig (2005) are shown in Table 1. The forward and inverse kinematics solutions were calculated in MATLAB (2019a, MathWorks, Mass., USA) using equations presented by Andersen (2018) and Hawkins (2013).

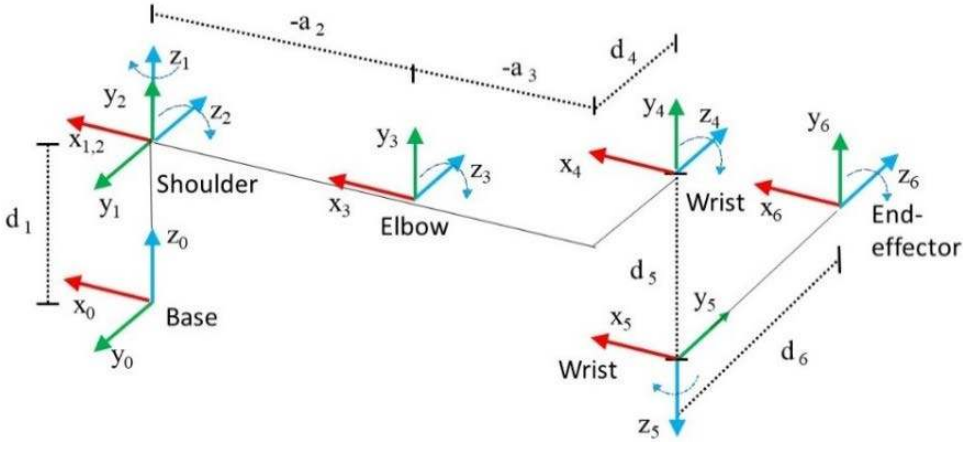

Figure 2. Coordinate frames description of UR5 manipulator

Table 1. Denavit-Hartenberg parameters for the UR5 manipulator (UR-Robotics, 2020)

Joints	Joint angle θ_i (rad)	Link length a_{i-1} (m)	Link offset d_i (m)	Link twist α_{i-1} (rad)
Joint 1 (Base)	θ_1	0	0.1625	π/2
Joint 2 (Shoulder)	θ_2	-0.425	0	0
Joint 3 (Elbow)	θ_3	-0.3922	0	0
Joint 4 (Wrist)	θ_4	0	0.1333	π/2
Joint 5 (Wrist)	θ_5	0	0.0997	-π/2
Joint 6 (End-effector)	θ_6	0	0.0996	0

The DH-parameters were used to calculate transformations between $i-1$ and i link using Eq. 1 for position of the end-effector. The inverse kinematic returns the set of joint angles (Eq. 3) based on a desired position and orientation of the tool link, specified as the transformation ${}^{i-1}_{i}T$ by satisfying Eq.2.

$$ {}^{i-1}_{i}T = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \sin\theta_i \cos(\alpha_{i-1}) & \cos\theta_i \cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) & -\sin(\alpha_{i-1})d_i \\ \sin\theta_i \sin(\alpha_{i-1}) & \cos\theta_i \sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & \cos(\alpha_{i-1})d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1) $$

$$ {}^{0}_{6}T = {}^{0}_{1}T(\theta_1)\,{}^{1}_{2}T(\theta_2){}^{2}_{3}T(\theta_3){}^{3}_{4}T(\theta_4){}^{4}_{5}T(\theta_5){}^{5}_{6}T(\theta_6) = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2) $$

$$ Q = [\theta_{1-6}] \quad (3) $$

Where, θ_i is the i^{th} joint angle, ${}^{0}_{6}T$ describes the desired transformations of the manipulator end-effector, Q represents the set of joint angles to reach the desired coordinate point (θ_{1-6}), and P_x, P_y, and P_z represents the position vector of the end-effector frame. Yadav and Garlanka (2018) solved the inverse kinematics of a 6R industrial robot by multiplying the transformation matrix with inverse transformation matrix to get a solvable equation. The inverse kinematic solutions were calculated as presented by Andersen (2018) for each joint angle (θ_i) (equations 4-9)

$$ \theta_1 = \text{atan2}({}^{0}P_{5y,}\,{}^{0}P_{5y}) \pm \text{acos}\left(\frac{d_4}{\sqrt{{}^{0}P_{5x}{}^{2}+{}^{0}P_{5y}{}^{2}}}\right) + \frac{\pi}{2} \quad (4) $$

$$ \theta_2 = \text{atan2}(-{}^{1}P_{4z}\,, -{}^{1}P_{4x}) - asin\left(\frac{-a_3 sin\theta_3}{|{}^{1}P_{4xz}|}\right) \quad (5) $$

$$ \theta_3 = \pm\,\text{acos}\left(\frac{|{}^{1}P_{4xz}|^2 - a_2{}^2 - a_3{}^2}{2a_2a_3}\right) \quad (6) $$

$$ \theta_4 = \text{atan2}\left({}^{3}\hat{X}_{4y}\,, {}^{3}\hat{X}_{4x}\right) \quad (7) $$

$$ \theta_5 = \pm\,\text{acos}\left(\frac{{}^{0}P_{6x}sin\theta_1 - {}^{0}P_{6y}cos\theta_1 - d_4}{d_6}\right) \quad (8) $$

$$ \theta_6 = \text{atan2}\left(\frac{-{}^{6}\hat{X}_{0y}.sin\theta_1 + {}^{6}\hat{Y}_{0y}cos\theta_1}{sin\theta_5}\,, \frac{{}^{6}X_{0x}.sin\theta_1 - {}^{6}Y_{0x}.cos\theta_1}{sin\theta_5}\right) \quad (9) $$

Where, ${}^{j}P_{i(x,y,z)}$ is the position of the i frame with respect to reference frame, θ_i is the i^{th} joint angle, namely the angle from X_{i-1} to X_i about Z_i, α_i is the angle from Z_i to Z_{i+1} about X_i , a_i is the distance from Z_i to Z_{i+1} along X_i, and d_i = distance from X_{i-1} to X_i along Z_i. The UR5 manipulator has multiple existing inverse kinematic solutions. The task space cubic polynomial trajectory solution was established based on the inverse solution involving minimum joint displacement to move the manipulator from the start to the desired position.

2.2 Virtual Tree Model

Canopy characteristics such as branch density and branch dimension can affect the path of the robotic manipulator to reach the object. The spatial requirements of the robotic manipulator to establish a collision-free trajectory can be estimated based on the location of branches. For a robotic pruning operation, the untargeted branches are considered as obstacles in the path of the robotic manipulator. These obstacles are usually varying in diameter, shape, size, and orientation. Tall spindle apple trees were targeted in this study. To simplify the tree structure, the tree trunk and branches were modelled in MATLAB using the cylinder bounding box function (Figure 3). The model was comprised of 13 obstacles, including one tree trunk and 12 primary branches, while the secondary and tertiary branches were not modelled for the study. Each branch was given a diameter, length and angle to get the position coordinates. The trunk and branches were labeled starting clockwise from the bottom to the top. The obstacle model was developed considering the manipulator workspace limitation that can effectively cover a section of a tree canopy. The depth of the tree canopy was set as 350 mm at each side of the trunk and the height was modelled as 600 mm starting from 500 mm above the ground surface. The position and diameter of the obstacle branches are shown in Table 2.

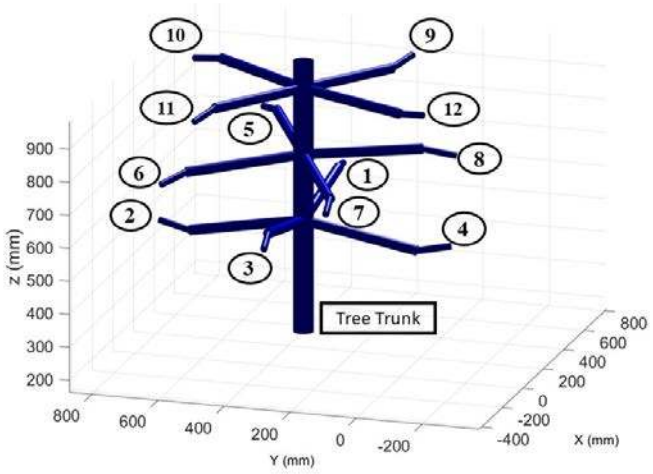

Figure 3. Virtual tree environment established in MATLAB, center bar is tree trunk, and 1-12 are branches

Table 2. Position and size parameters of the branches

Branch No.	Diameter (mm)	Starting point coordinates			Terminal point coordinates		
		x (mm)	y (mm)	z (mm)	x (mm)	y (mm)	z (mm)
1	15	-355	415	513	338	844	448
2	18	-370	402	513	-789	282	533
3	14	-355	430	513	-324	-40	451
4	16	-340	402	513	92	408	539
5	17	-334	410	712	-604	758	647
6	14	-348	394	712	-746	205	643
7	17	-363	440	712	-282	-42	715
8	18	-348	394	712	33	630	716
9	15	-351	415	914	-137	792	885
10	14	-370	397	914	741	617	893
11	15	-351	382	914	-510	-19	902
12	13	-337	397	914	43	182	900

2.3 Integrated Simulation Environment

A virtual environment was established in MATLAB to perform simulation for branch accessibility. The kinematic model of the manipulator (Kutzer, 2020) and the developed virtual tree were imported to the simulation environment. The workspace of the UR5 manipulator was calculated as shown in Figure 4(a) to estimate the working environment for simulation. The base of the manipulator was set as origin i.e. x= 0 mm, y=0 mm, and z= 0 mm. Based on the allowable workspace of the manipulator and the physical parameters of virtual tree model, the distance between the manipulator base and tree trunk was set at 400 mm (Figure 4(b)). The orientation of the virtual tree was set to make most branches accessible to the manipulator.

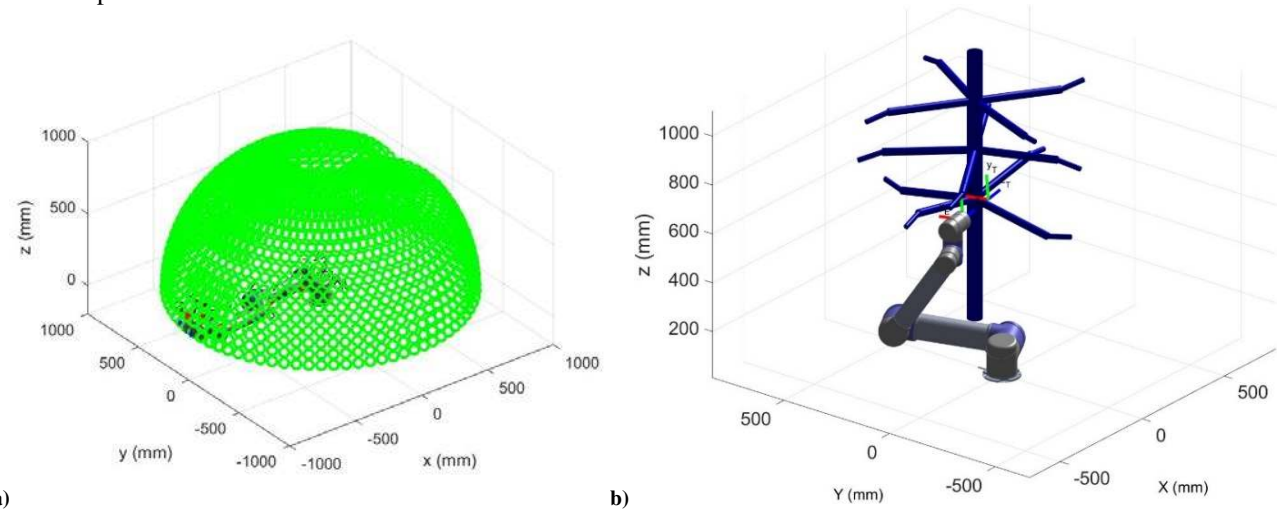

Figure 4. (a) Workspace envelop of the robotic manipulator, and (b) Virtual environment setup in MATLAB

3. Path Planning Algorithm Overview

3.1 RRT Path Planning

The RRT algorithm searches and constructs an object using uniform random sampling in search space. RRT works in a configuration space, which includes all transformations applied to the manipulators (LaValle, 2006). The pseudocode and details of major functions for RRT algorithm are shown in Table 3. To address the collision free path problem in an obstacle environment, the given configuration space is expressed as $K \subset R^n$ ($n \in N$), where n refers to dimensions for a given space. The area occupied by the obstacle in the space is described as $K_{obs} \subset K$ and the area without any obstacle is described as $K_{free} = K / K_{obs}$. The target is represented as $k_{target} \subset K_{free}$ and start point is represented as $k_{initial} \subset K_{free}$. The nodes $k_{initial}$ and k_{target} inputs for the path planner and the aim is to find the obstacle free path from $k_{initial}$ to k_{target} in K_{free} search space in a predefined fixed iteration. As shown in Figure 5, the tree exploration starts from an initial point ($k_{initial}$) and expands gradually as the process iteratively continues seeking the target point (k_{target}). Starting from the $k_{initial}$ as the root, a random state (k_{rand}) is selected in the search space for each iteration. The algorithm search for all the expanded nodes of the tree to obtain a node nearest ($k_{nearest}$) to k_{rand}. If the k_{rand} exists in obstacle free in search space, a nearest node $k_{nearest}$ is searched according to the defined step size (ρ). Otherwise, the steering function adds a new node k_{new}, and the tree explanation is expanded by connecting $k_{nearest}$ to k_{new}. The node k_{new} is added with a defined step size to expand the tree. When the manipulator is in k_{new}, it searches for a collision between the manipulator and obstacles. If a collision exists, the node k_{new} is discarded. If there is no collision, k_{new} is added to expand the tree exploration. For every k_{new}, the distance between k_{new} and the target point (k_{target}) is compared with the set threshold. If the calculated distance is less or equal to the set threshold, the solution is found. The path solution is calculated by tracking back from the k_{target} to start point.

Table 3. Pseudocode and major functions for basic RRT algorithm

$T = (V, E) \leftarrow RRT(k_{init.})$	Details of major functions
1. $T \leftarrow$ Initialize Tree (); *2. $T \leftarrow$ InsertNode(Ø, $k_{init.}$, T);* *3. for i =0 to i=N do* *4. $k_{rand} \leftarrow$ Random_state()* *5. $k_{nearest} \leftarrow$ Nearest_neighbor(T, k_{rand})* *6. (k_{new}, U_{new}) $\leftarrow$ Steer ($k_{nearest}$, k_{rand});* *7. if Obstaclefree(k_{new}) then* *8. $T \leftarrow$ InsertNode(k_{min}, k_{new}, T);* *9. return T* *10. end*	*Random State: Generate a random node k_{rand} in obstacle free region k_{free} in configuration space (K)* *Nearest: Generates $k_{nearest}$ to k_{rand} from T = (V, E) defined by a cost function* *Steer: Generate control input u [0, T] to drive k(0)= k_{rand} to k(T)= $k_{nearest}$ along the path k:[0,T] → K adding k_{new} at incremental distance (Δq) from $k_{nearest}$ towards k_{rand}* *CollisionCheck: Check collision and returns true if tree is collision free, i.e., k:[0, T] is k_{free} at t=0 to t=T* *InsertNode: Adds a node k_{new} to V in order to connect node k_{min} as its parent in the tree T = (V, E).*

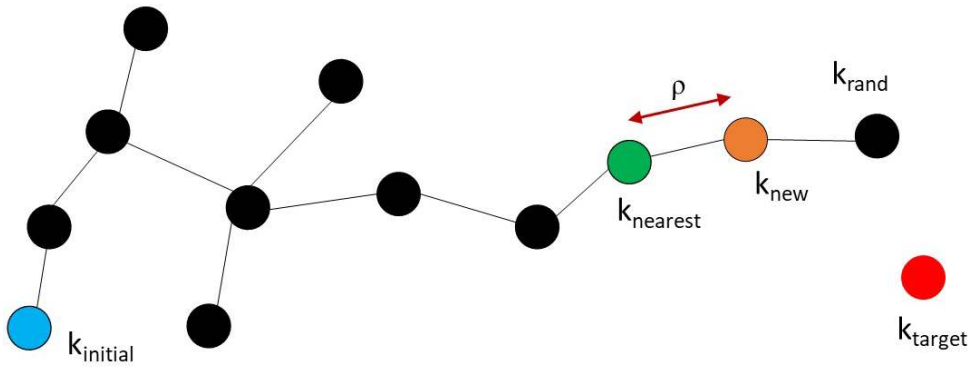

Figure 5. Illustration of RRT exploration

Figure 6 shows the process flow chart of RRT path finding operation in the given obstacle model. Once the obstacle model is setup within the workspace of the manipulator, the path finding process starts with establishing boundary cylinders around each link of the manipulator to check for collisions associated with a body. The coordinates of target points are inputted to initialize the RRT exploitation according to the defined step size and variables. At that point, the RRT algorithm performs two checks, manipulator body collision and path collision. For the former, the RRT algorithm performs the manipulator side collision check with the obstacles and the links self-collision check using the boundary cylinders developed around each link. The side-interactions and end-interactions of the boundary cylinders returns collision with branch and self-collision of the manipulator body, respectively. If the algorithm returns no collision of boundary cylinders, the specific nodes are added to the path solution, and process continues until connected nodes reach the target location. Similarly, algorithm search for a collision-free path for the movement of integrated end-effector by connecting multiple free nodes in the search space, and the process continues until the target location or allowed iterations are achieved. The same process is repeated until the path for all defined target points are calculated.

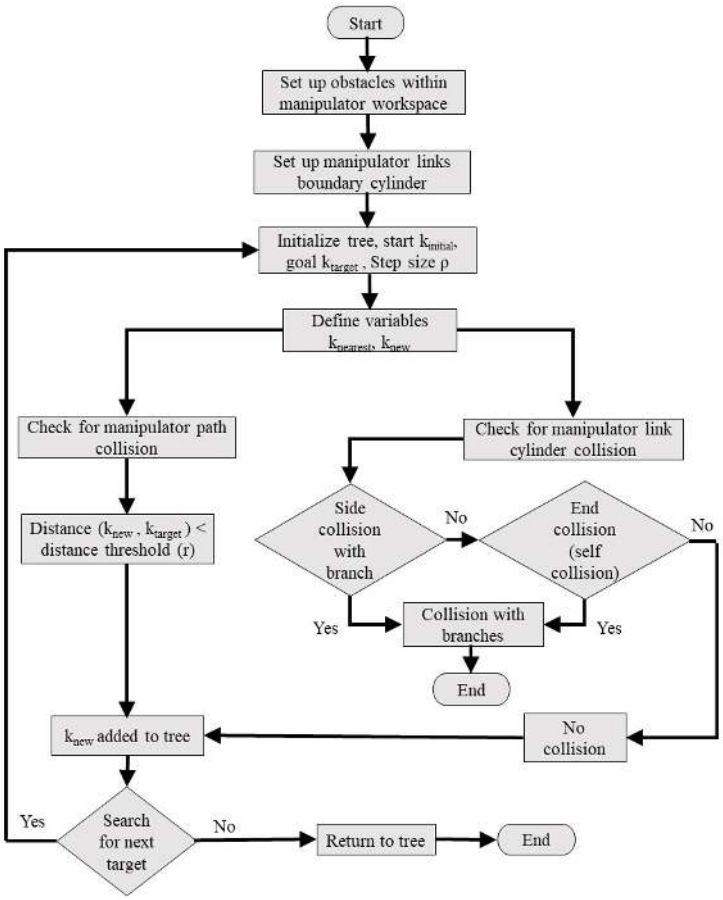

Figure 6. Flow chart for RRT collision free path planning

3.2 Path Smoothing and Optimization Algorithms

The path solution calculated by RRT may not be the shortest and smoothest path to reach the target. It had some redundant and unnecessary nodes which increased the path length. To address this issue, a smoothing method presented by Cao et al. (2019) was combined with the RRT path to remove the unnecessary nodes to shorten the path length. As RRT utilizes a random sampling technique within the search space, the nodes are always connected in an arbitrary direction to create the collision free path. However, some of these connected nodes were unnecessary, and removing those will not affect the collision free selection of nodes. Figure 7 illustrates this node elimination process. For example, in a given search space, the exploration tree connects nodes *n* to *n+i* to establish a path (red line) avoiding two obstacles. The nodes *n* and *n+2* can be connected directly with a displacement vector (green dashed line) suppressing the unnecessary node *n+1*, while the path remains collision free. This process results in shorter path length to move between node *n* to *n+2* and, smooth the manipulator movement. This method of removing unnecessary redundant nodes was implemented for the whole RRT path, then a new shorter and smoother path was obtained.

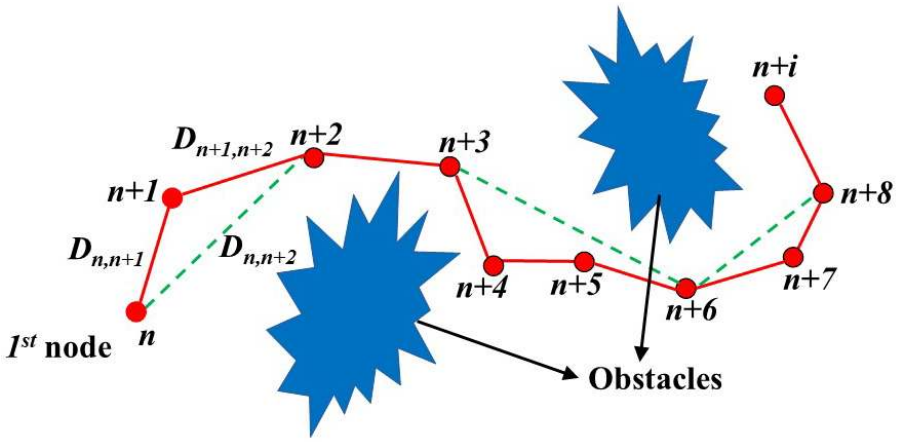

Figure 7. Illustration of RRT path smoothing

The process chart of the RRT path smoothing is shown in Figure 8. As the RRT path was calculated by connecting '*(n)+(n+i)*', where *n* is the starting node, *n+i* are the subsequent nodes. The RRT path begins with node *n* at the initial position, *n+1* is the 2nd node, *n+2* is the 3rd node, *n + 3* is the 4th node, and it goes on until it reaches the end point of the path by connecting *i* nodes. The smoothing process begins once all the nodes generated from RRT are added to the path. The algorithm was set to consider three consecutive nodes at any instance e.g. *n*, *n+1,* and *n+2.* As the algorithm works on the displacement vector (shortest path length), the distances between nodes *n* to *n+1, n+1* to *n+2,* and *n* to *n+2* were calculated and referred as $D_{n,n+1}$, $D_{n+1,n+2}$, and $D_{n,n+2}$ respectively. Based on the general rule of displacement vector which refers to shortest distance between two nodes, if the $D_{n,n+2}$ was less than the sum of $D_{n,n+1}$ and $D_{n+1,n+2}$, the node *n* was directly connected to *n+2.* A check was also performed to detect the obstacles in the direct path between *n* and *n+2*. For example, considering moving from node *n+2* to node *n+5*, the node *n+3* and node *n+4* could be eliminated from the path based on the displacement rule, but these nodes cannot be connected directly as there is a collision in a the direct path from node *n+2* to node *n+5*. To perform the collision check, the additional collision detection points were added between newly connected nodes *n* and *n+2* at a distance less than the step size (ρ) originally used to find the RRT path. If all the new detection points return k_{free} in configuration space, the smooth path was marked as collision free and node *n+1* was eliminated. The process was repeated for all other nodes making a set of three nodes e.g. *n*, *n+2,* and *n+3* until *n+i* was covered. If any of detection point return k_{obs}, the displacement vector between the *n* to *n+2* was discarded to keep original RRT path i.e. *n* to *n+1* to *n+2* and the next process was repeated for *n+2, n+3,* and *n+4* until *n+i* was covered. The new path generated using this method was referred to as a smooth path.

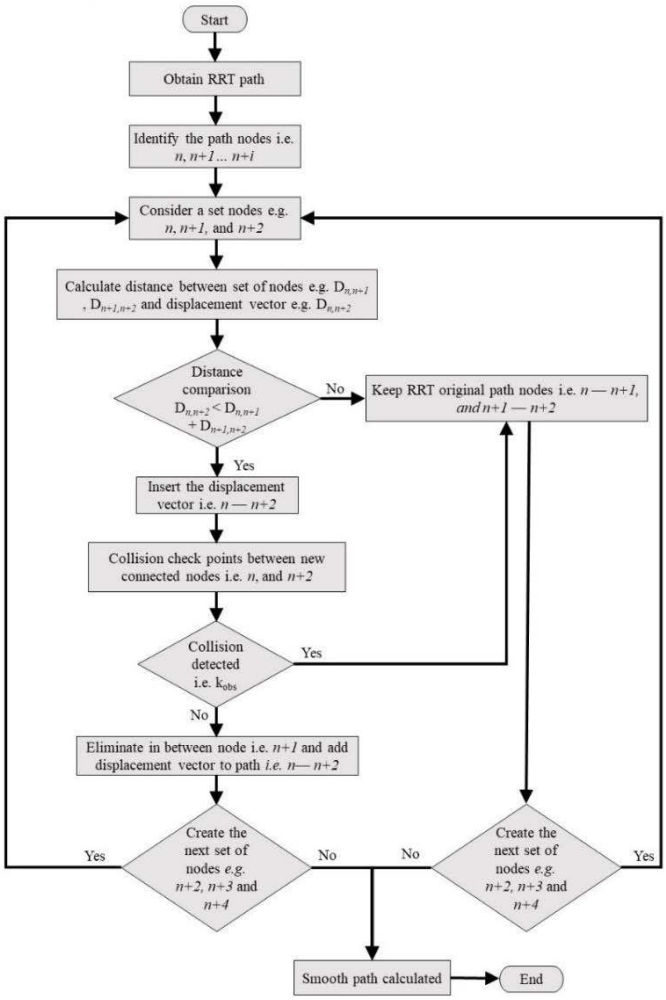

Figure 8. Flowchart for RRT path smoothing

Finally, the optimization of path finding was performed using the MATLAB optimization toolbox. For a given target position, the toolbox function provides a collision free optimized path solution in the search space. A non-linear optimization function *fmincon*() was used to obtain the optimized path solutions. The two basic indicators were set with the *fmincon*(), i.e. cost or objective function, and constraint to calculate the optimized path. The cost is the function whose result is desired to be minimal. For this study, the objective was to minimize the total length of the path or the traveled distance from the start to the target position coordinates. The constraint function gives the final collision free path based on the set threshold for minimum distance between the end effector path and manipulator links to the obstacles in the search space. The minimum distance threshold between the optimized path and obstacles, and manipulator links and obstacles, was set as 60 mm. For optimization algorithms, sometimes the output is confined in a local minimal and not the optimal one, thus the initial values

were assigned to obtain an optimal solution. The workflow of the optimization process includes establishing a lower and upper boundary, generation of the initial values, and the cost and constraint functions. The optimized path was calculated, and a cubic polynomial trajectory was generated for the manipulator to reach the target location. The process is repeated to find the optimized path for next target position until the path was calculated for all the target branches in the scene.

4. Simulation for Branch Accessibility

Simulation for collision free path planning for UR 5 manipulator was performed in the developed MATLAB virtual environment. The simulation setup was established on a computer system with Windows 10 operating system (Microsoft Inc., USA) and 16 GB memory. The simulation was performed for reaching different target pruning points in the virtual tree. The coordinates of target pruning points on each branch were marked 50 mm away from the tree trunk. The target orientation and approach pose of the end-effector was kept the same to compare the performance of each path planning method. For each simulation run, three branches were selected as target pruning branches in the scene. The data were recorded for path finding success, path length, and computational time for all path planning method to compare the performance. Considering the randomness of RRT path planning as it generates different set of nodes during each trial, the simulation was performed ten times to calculate the mean path length for the same branches in the scene. The standard deviation and coefficient of variance (*cv*) of the mean path length were calculated to verify the repeatability of the process. The following simulation was performed:

- 1st simulation: To test the performance of RRT path planning.
- 2nd simulation: To test the effectiveness of RRT path smoothing method.
- 3rd simulation: Applying optimization method to find optimal path solution.

The results of these simulation methods were also calculated separately to test the performance in terms of path finding success, path length, and processing time for each method. The algorithm stops the manipulator for 2 seconds at each target position, which represents the time required to make the actual cut.

5. Results and Discussion

Branch accessibility was simulated for three target pruning branches: branch 2, 8, and 6. The simulation was conducted at the same cutter pose at the target, i.e., perpendicular to the branch in the horizontal plane. The manipulator started from the home position and returned to the home position after simulating the path for each targeted point. Branch 2 and 8 have one obstacle in the path while branch 6 has two obstacle branches in the path. The results for mean values and statistical calculations of path length, computational time for RRT path and path with smoothing method, and optimized path are shown in Table 4.

Table 4. Simulation results for RRT path planning and smoothing for different target branches

Target branch	No. of obstacles	End-effector approach pose with no rotation i.e. ideal cutting pose								
		RRT			RRT with smoothing			Path optimization		
		Path length (mm)	Computation time (sec)	Move time (sec)	Path length (mm)	Computation time (sec)	Move time (sec)	Path length (mm)	Computation time (sec)	Move time (sec)
2	1	390	19	9	257	21	7	278	36	8
8	1	*	*	*	*	*	*	*	*	*
6	2	496	25	14	382	29	12	397	24	13

	Statistical calculations for path length			
	Branch 2		Branch 6	
	Standard Deviation	*Variation Coefficient (cv)*	*Standard Deviation*	*Variation Coefficient (cv)*
RRT	26	0.066	28	0.058
Path smoothing	11	0.042	15	0.039
Path optimization	10	0.037	12	0.031

*Failed to find the collision free path

During the simulation, collision-free paths were obtained for branch 2 and 6. The RRT algorithm successfully avoided one obstacle i.e. branch 3 for collision free path to branch 2, and similarly avoided the two obstacles i.e. branch 3 and 7 to create the collision free path for branch 6. The mean path length for target points was calculated as 390 and 496 mm for reaching branch 2 and branch 6 respectively, and the corresponding computing time was calculated as 19 and 25 seconds. For branch 8, the algorithm did not successfully find a path solution because at a normal pose approach, the wrist (joints 4 and 5) and the end-effector occupy a horizontal length of 300 mm to attain the desired posture but the cylinder coordinates of branch 7 was present in the environment, thus there exists no possible solution to establish the collision free path for branch 8 at the normal pose approach. The RRT smoothing simulation was also performed to reduce the path length for the same set of branches. The mean path lengths for branch 2 and 6 were calculated as 257 and 382 mm respectively, and the processing time was calculated as 21 and 29 seconds respectively while no path was obtained for branch 8 due to obstacle

constraints. The mean smooth path lengths for branches 2 and 6 were less by an amount of 34% and 22% respectively compared to the original RRT path lengths. Similarly, the optimization method was used to simulate the paths for the same scene. Branch 8 was found not accessible due to the complexity of the surrounding environment while the mean path lengths for branch 2 and 6 were calculated as 278 and 397 mm respectively. The time was calculated as 36, and 24 seconds respectively. The result showed that the mean optimized path lengths for branch 2 and 6 were less compared to RRT path planning by 29% and 20% respectively and larger compared to RRT smooth path by 7% and 4% respectively. The performance with path optimization showed better results in terms of computational time and path length compared to RRT path planning. The mean path lengths and computation time for all branches using the optimization method was less than the RRT path, while less difference was observed with the RRT smooth path. The success rate was similar for all three methods, but the computational time and path length were found to be different. The standard deviation and variation coefficient (cv) were calculated to confirm the repeatability of each process. For all branch 2 and 6 tests, the standard deviation for RRT (26, and 28 respectively) was higher compared to RRT smoothing (14 and 17 respectively), and optimization method (10 and 12 respectively) which is due to the randomness associated with RRT tree exploration. The cv for branch 2 and 6 was the lowest value, 0.037 and 0.031 respectively. This was recorded for the path calculated using the optimization function which confirms the repeatability of the process compared to the other two methods. For all branches, the shortest mean path lengths were calculated using the RRT with smoothing method, and elapsed time of optimization function was observed lowest for most branches.

Figure 9(a-c) shows the path planning of the robotic manipulator from the initial position to branch 6 using the three different path planning methods. The red line in Figure 9 (a) shows the RRT path established by connecting multiple random nodes. The randomness of the tree exploration increased the path length and affects path smoothness. Although the path is collision free, it has many unnecessary nodes. The next simulation was performed for RRT path planning combined with path smoothing. The path smoothing algorithm successfully eliminated the redundant nodes and reduced the path length. The green path line in Figure 9 (b) shows the smoothed path for branch 6. The mean RRT smooth path length was calculated as 382 mm which is approximately 22% less than the mean RRT path length. The total time for combined RRT smoothing was not different from the RRT path. The computational time for RRT smooth path generation was an additional 1 to 3 seconds compared to RRT path, which was compensated by the lesser manipulator trajectory time, as a shorter path length requires less time to reach the target point. The blue line in Figure 9 (c) shows the optimized path for branch 6. The length of optimized path was less than RRT path, but only a small difference was observed for the RRT smooth path.

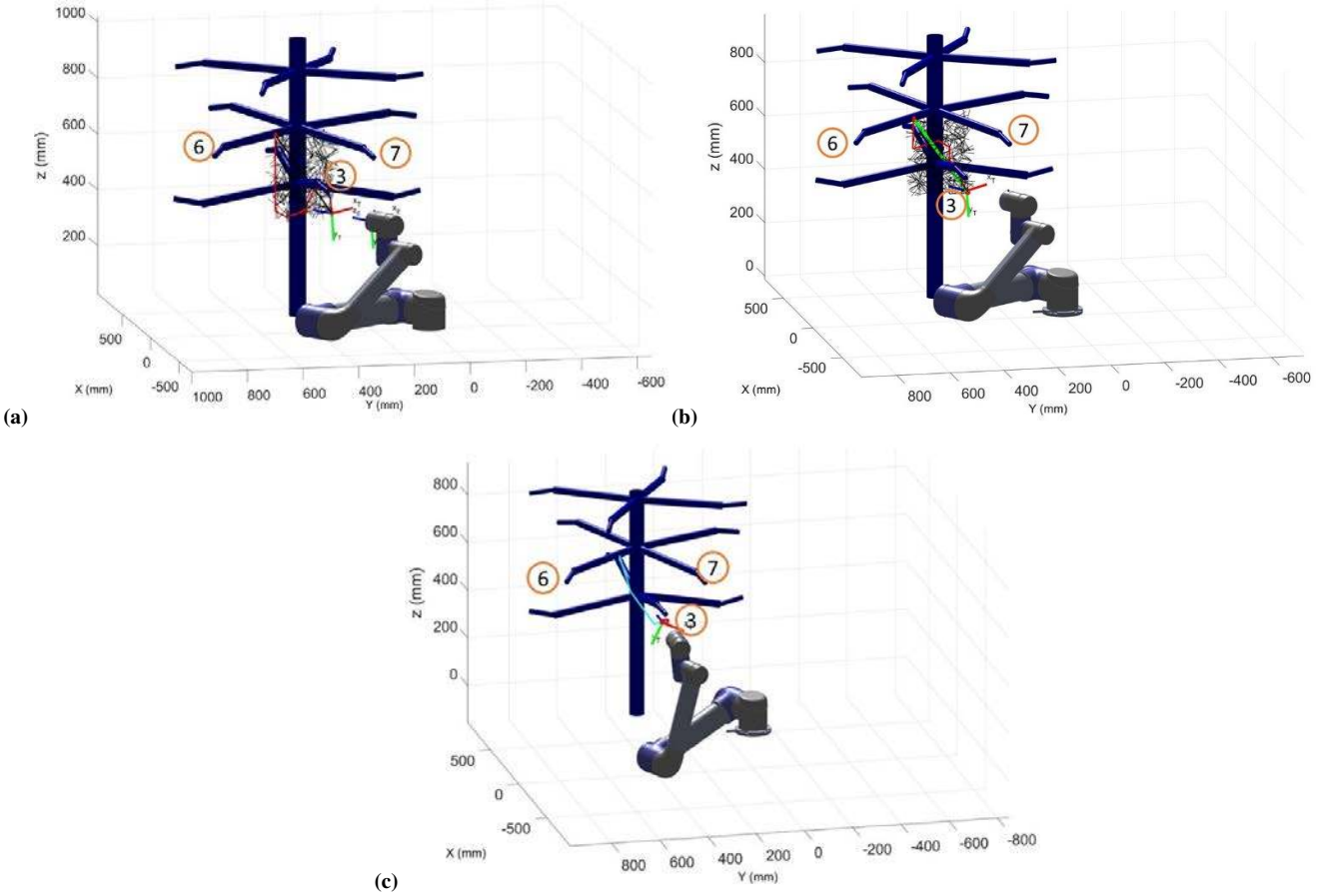

Figure 9. Path planning for branch 6. (a) RRT path (red line), (b) RRT smooth path (green line), and (c) Optimized path (blue line)

The path lengths and computational time depends on many factors including number of obstacles, distance between start and target, step size, minimum distance threshold, and repeatability of RRT random sampling in the search space. To have a safe path from start to target position, every point in the search space should be detected to avoid collisions. The role of step size is critical for safe path planning. The step size and probability of manipulator collision are directly related to each other but inversely related to computational time. The smaller step size reduced the probability of collision, but the computational time was high and vice versa. Therefore, the decision on step size and minimum distance from obstacles depends on the required degree of path safety as well as allowable computation time for each target point. For all methods, the computation time was observed different for all three selected branches as the search space size, and number and position of obstacles were different. For example, branch 6 has two obstacles in the path, therefore the computational time was much higher, 29 seconds for RRT smooth path. Meanwhile, branch 6 was the farthest from the manipulator start position, which meant that it had a large search space for RRT tree exploration as well as requiring more time for manipulator trajectory. The path smoothing time was not different for all target points as the search space for path smoothing was a narrow region very close to the RRT path which is already in a collision free search space, therefore less computational time was required to perform additional RRT path smoothing.

The algorithm developed successfully created a collision free path planning in the virtual tree model. Although the RRT successfully avoided the obstacles to generate a trajectory for reaching the target position at the desired cutter orientation, the path length and computational time usually vary due to the randomness associated with RRT exploration. For multiple simulation trials of the same branch, the lowest elapsed time was observed corresponding to the largest RRT path length. For example, during different trials for branch 6, the path lengths were 617 and 515 mm with elapsed time 19 and 27 seconds respectively. The generated path is larger because exploration was directed towards the target and nodes farthest from obstacles could connect quickly due to absence of obstacles in the search space. For shorter paths, the exploration is directed towards the target and as a result, more time is required to establish a path in the obstacle workspace. The smoothing method used in the study has reduced the RRT path lengths, making it nearly equal to the optimized path found by using optimization toolbox functions. Also, RRT randomness could possibly return an entirely different path for the same branch. For example, to reach branch 6, nearly half of the simulation trials established a path above the obstacle branch 3 as shown in Figure 9 (b), while in other simulation trials, the path was created below the obstacle branch 3 as shown in Figure 9 (a). Therefore, in addition to the consideration for varied path lengths, the stability and repeatability for path generation should also be considered. In the future, to reduce the risk for path variation, the modified version of RRT algorithms such as RRT star (RRT*), and RRT-smart (RRT-S) along with optimization algorithms such as Genetic algorithm (GA) will be implemented to direct the RRT exploration towards the target. Furthermore, the obstacle model was simplified for parameters such as diameter, shape, position, orientation, and position of branches. The algorithm works well for the adopted obstacle model but for a more complex real world tree that includes multiple secondary branches at wide orientation ranges, the algorithm may need some adjustment based on the physical parameters of the apple tree canopy as well as environmental parameters including wind speed and terrain. The orientation of the end-effector cutter tool is also critical in robotic pruning. Although, the limb renewal is not influenced by the bevel or straight cut (Schupp et al., 2019), but the cutter plane (yz_T) of robotic pruner should be aligned with branch axis for positioning the branch in the cutter opening/pivot. The selection of cutter orientation is not only based on the orientation of the target branches but also on the orientation and position of the nearby obstacles.

6. Conclusions

The simulation for branch accessibility with a 6R DoF robotic manipulator was performed to establish a collision-free trajectory for reaching target pruning locations. A virtual environment was established for path planning. The following conclusions were drawn from the study.

1. The RRT algorithm was successful in finding a collision-free path for defined pruning points within the virtual tree environment. The RRT path may include some retention nodes which increases the path length.
2. The smoothing method successfully reduced the RRT path lengths for all target branches by removing the redundant nodes in the original path.
3. The variation coefficient of path length for optimization method was low compared to RRT with smoothing, which confirms the stability and repeatability of the method. However, the smallest path length was achieved using the RRT with smoothing method.
4. The developed RRT algorithm was relatively slow in path finding. A modified RRT is suggested with optimization algorithms to stabilize the path generation and improve the obstacle avoidance efficiency. Also, the step size for tree exploration should be optimized considering the required accuracy of collision avoidance and computational time.

In the future, lab tests will be conducted to validate the results using a UR5 robotic manipulator integrated with a modified shear pruner end-effector. A vision system will be integrated to create a 3D obstacle model of the real-world apple tree branches and the collision-free path will be simulated. These future studies will lay the foundation for the development of robotic pruning system for economical and sustainable apple production system.

Acknowledgements

This research was partially supported in part by United States Department of Agriculture (USDA)'s National Institute of Food and Agriculture Federal Appropriations under Project PEN04547 and Accession No. 1001036. We also would like to give our special thanks for the support from Penn State College of Agricultural Sciences Stoy G. and Della E. Sunday program and Northeast Sustainable Agriculture Research and Education (SARE) Graduate Student Grant GNE19-225-33243.

References

Andersen, R. S. (2018). Kinematics of a UR5. In *Aalborg University, May 2018*. Retrieved from http://rasmusan.blog.aau.dk/files/ur5_kinematics.pdf

Bac, C. W., Henten, E. J., Hemming, J., & Edan, Y. (2014). Harvesting robots for high-value crops: state-of-the-art review and challenges ahead. *Journal of Field Robotics*, *31*(6), 888–911. https://doi.org/10.1002/rob.21525

Cao, X., Zou, X., Jia, C., Chen, M., & Zeng, Z. (2019). RRT-based path planning for an intelligent litchi-picking manipulator. *Computers and Electronics in Agriculture*, *156*, 105–118. https://doi.org/10.1016/j.compag.2018.10.031

Carbone, G., Ceccarelli, M., Oliveira, P., Saramago, S., & Carvalho, J. (2008). An optimum path planning for cassino parallel manipulator by using inverse dynamics. *Robotica*, *26*, 229–239. https://doi.org/10.1017/S0263574707003839

Choi, Y.-K., Park, J.-H., Kim, H.-S., & Kim, J. (2000). Optimal trajectory planning and sliding mode control for robots using evolution strategy. *Robotica*, *18*, 423–428. https://doi.org/10.1017/S0263574799002118

Craig, J. (2005). *Introduction to robotics: mechanics and control* (3rd ed.). https://doi.org/10.7227/ijeee.41.4.11

Elbanhawi, M., & Simic, M. (2014). Sampling-based robot motion planning: a review. *IEEE Access*, *2*, 56–77. https://doi.org/10.1109/ACCESS.2014.2302442

Goerzen, C., Kong, Z., & Mettler, B. (2009). A survey of motion planning algorithms from the perspective of autonomous UAV guidance. *Journal of Intelligent and Robotic Systems*, *57*(1), 65. https://doi.org/10.1007/s10846-009-9383-1

Gómez-Bravo, F., Carbone, G., & Fortes, J. C. (2012). Collision free trajectory planning for hybrid manipulators. *Mechatronics*, *22*(6), 836–851. https://doi.org/10.1016/J.MECHATRONICS.2012.05.001

Gongal, A., Silwal, A., Amatya, S., Karkee, M., Zhang, Q., & Lewis, K. (2016). Apple crop-load estimation with over-the-row machine vision system. *Computers and Electronics in Agriculture*, *120*, 26–35. https://doi.org/https://doi.org/10.1016/j.compag.2015.10.022

Hawkins, K. P. (2013). Analytic inverse kinematics for the Universal Robots UR-5/UR-10 arms. Retrieved March 3, 2020, from Georgia Tech, Dec 2013 website: http://hdl.handle.net/1853/50782

He, L., & Schupp, J. (2018). Sensing and automation in pruning of apple trees: a review. *Agronomy*, *8*(10), 211. https://doi.org/10.3390/agronomy8100211

Kapach, K., Barnea, E., Mairon, R., Edan, Y., & Shahar, O. Ben. (2012). Computer vision for fruit harvesting robots: state of the art and challenges ahead. *International Journal of Computational Vision and Robotics*, *3*(1/2), 4. https://doi.org/10.1504/IJCVR.2012.046419

Karaman, S., & Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, *30*(7), 846–894. https://doi.org/10.1177/0278364911406761

Karkee, M., Adhikari, B., Amatya, S., & Zhang, Q. (2014). Identification of pruning branches in tall spindle apple trees for automated pruning. *Computers and Electronics in Agriculture*, *103*, 127–135. https://doi.org/10.1016/j.compag.2014.02.013

Kondo, N., Shibano, Y., Mohri, K., & Monta, M. (1993). Basic studies on robot to work in vineyard 1: Manipulator and harvesting hand. *J. Jpn. Soc. Agric. Mach.*, *55*, 85–94.

Kondo, N., Shibano, Y., Mohri, K., & Monta, M. (1994). Basic studies on robot to work in vineyard 2: Discriminating, position detecting and harvesting experiments using a visual sensor. *J. Jpn. Soc. Agric. Mach.*, *56*, 45–53.

Kuffner, J. J., & La Valle, S. M. (2000). RRT-connect: an efficient approach to single-query path planning. *Proceedings - IEEE International Conference on Robotics and Automation*, *2*, 995–1001. https://doi.org/10.1109/robot.2000.844730

Kurfess, T. R. (2005). *Robotics and Automation Handbook* (1st ed.). Retrieved from https://doc.lagout.org/science/0_Computer Science/8_Electronics %26 Robotics/Robotics and Automation Handbook.pdf

Kutzer, M. (2020). Kutzer/URToolbox. Retrieved March 7, 2020, from https://github.com/kutzer/URToolbox

LaValle, S. M. (1998). Rapidly-Exploring Random Trees: a new tool for path planning. *Iowa State University Ames, IA 50011 USA*. https://doi.org/10.1063/1.5030320

LaValle, S. M. (2006). Planning algorithms. In *University of Illinois* (1st ed.). https://doi.org/10.1017/CBO9780511546877

Lehnert, R. (2012). Robotic pruning. Retrieved April 25, 2019, from Good Fruit Grower Nov. 1, 2012 website: https://www.goodfruit.com/robotic-pruning/

Li, P., Lee, S. H., & Hsu, H. Y. (2011). Review on fruit harvesting method for potential use of automatic fruit harvesting systems. *Procedia Engineering*, *23*, 351–366. https://doi.org/10.1016/j.proeng.2011.11.2514

Lin, C., Chang, P., & Luh, J. (1983). Formulation and optimization of cubic polynomial joint trajectories for industrial robots. *IEEE Transactions on Automatic Control*, *28*(12), 1066–1074. https://doi.org/10.1109/TAC.1983.1103181

Lindner, M., Kolb, A., & Hartmann, K. (2007). Data-fusion of PMD-based distance-information and high-resolution RGB-images. *2007 International Symposium on Signals, Circuits and Systems*, *1*, 1–4. https://doi.org/10.1109/ISSCS.2007.4292666

Nakarmi, A. D., & Tang, L. (2012). Automatic inter-plant spacing sensing at early growth stages using a 3D vision sensor. *Computers and Electronics in Agriculture*, *82*, 23–31. https://doi.org/10.1016/J.COMPAG.2011.12.011

Nasir, J., Islam, F., Malik, U., Ayaz, Y., Hasan, O., Khan, M., & Muhammad, M. S. (2013). RRT*-SMART: a rapid convergence implementation of RRT*. *International Journal of Advanced Robotic Systems*, *10*, 299. https://doi.org/10.5772/56718

Noreen, I., Khan, A., & Habib, Z. (2016a). A comparison of RRT, RRT* and RRT*-Smart path planning algorithms. *International Journal of Computer Science and Network Security*, *16*(10), 20–27. Retrieved from http://paper.ijcsns.org/07_book/201610/20161004.pdf

Noreen, I., Khan, A., & Habib, Z. (2016b). Optimal path planning using RRT* based approaches: a survey and future directions. *International Journal of Advanced Computer Science and Applications*, *7*(11), 97–107. https://doi.org/10.14569/ijacsa.2016.071114

Saramago, S., & Ceccarelli, M. (2004). Effect of basic numerical parameters on a path planning of robots taking into account actuating energy. *Mechanism and Machine Theory*, *39*, 247–260. https://doi.org/10.1016/S0094-114X(03)00117-4

Schupp, J. R., Winzeler, H. E., & Schupp, M. A. (2019). Stub length and stub angle did not influence renewal shoot number or branch angle of tall spindle 'Gala'/Malling 9 apple trees. *HortTechnology Hortte*, *29*(1), 46–49. https://doi.org/10.21273/HORTTECH04218-18

Silwal, A. (2016). *Machine vision system for robotic apple harvesting in fruiting wall orchards*. Washington State University.

UR-Robotics. (2020). Parameters for calculations of kinematics and dynamics - 45257. Retrieved March 3, 2020, from Universal Robots website: https://www.universal-robots.com/how-tos-and-faqs/faq/ur-faq/parameters-for-calculations-of-kinematics-and-dynamics-45257/

USDA-NASS. (2019). Noncitrus Fruits and Nuts 2018 Summary, United States Department of Agriculture - National Agricultural Statistics Service. In *Washington, DC: USDA-NASS,*. https://doi.org/Retrieved fromhttps://www.nass.usda.gov/Publications/Todays_Reports/reports/ncit0619.pdf

Van Henten, E. ., Hemming, J., Van Tuijl, B. A. ., Kornet, J. ., & Bontsema, J. (2003). Collision-free motion planning for a cucumber picking robot. *Biosystems Engineering*, *86*(2), 135–144. https://doi.org/10.1016/S1537-5110(03)00133-8

Vision Robotics Corporation. (2015). Retrieved October 12, 2018, from Intelligent Autonomous Grapevine Pruner website: https://www.visionrobotics.com/vr-grapevine-pruner

Wu, D., Sun, Y., Wang, X., & Wang, X. (2016). An improved RRT algorithm for crane path planning. *International Journal of Robotics and Automation*, *31*(2), 84–92. https://doi.org/10.2316/Journal.206.2016.2.206-4180

Yadav, B., & Garlanka, S. B. (2018). Trajectory planning of 6R jointed industrial robot manipulators. *International Journal for Research in Applied Science and Engineering Technology*, *6*(6), 1760–1766. https://doi.org/10.22214/ijraset.2018.6259

Yang, H., Li, L., & Gao, Z. (2017). *Obstacle avoidance path planning of hybrid harvesting manipulator based on joint configuration space*. *33*, 55–62. https://doi.org/10.11975/j.issn.1002-6819.2017.04.008