BOHR International Journal of Internet of things,
Artificial Intelligence and Machine Learning
2022, Vol. 1, No. 1, pp. 1–11
DOI: 10.54646/bijiam.2022.01
www.bohrpub.com

METHODS

Robotic mushroom harvesting by employing probabilistic road map and inverse kinematics

M. G. Mohanan* **and Ambuja Salgaonkar**

Department of Computer Science, University of Mumbai, Mumbai, India

***Correspondence:**
M. G. Mohanan,
mgmohanam@gmail.com;
mohanan@udcs.mu.ac.in

Received: 04 February 2021; **Accepted:** 06 January 2022; **Published:** 08 February 2022

A collision-free path to a destination position in a random farm is determined using a probabilistic roadmap that can manage static and dynamic obstacles. The position of ripening mushrooms is a result of picture processing. A mushroom harvesting robot is explored that uses inverse kinematics at the target position to compute the state of a robotic hand for grasping a ripening mushroom and plucking it. The Denavit–Hartenberg approach was used to create a kinematic model of a two-finger dexterous hand with three degrees of freedom for mushroom picking. Unlike prior experiments in mushroom harvesting, mushrooms are not planted in a grid or design but are randomly scattered. At any point throughout the harvesting process, no human interaction is necessary.

Keywords: mushroom harvesting, probabilistic road map, motion planning, inverse kinematics

Introduction

This provides pointers to contemporary research on employing robotics in agriculture. The emphasis is on automation of the process of mushroom plucking. The method is applicable for the plucking of mushrooms if they are cultivated in a bed or even otherwise, if they are cultivated randomly on a farm. Many times, mushrooms are ready but there is little skilled labor available to pluck them; this situation causes a significant loss to mushroom farmers because the mushrooms perish immediately. A dexterous robotic hand model proposed in this paper is a solution to overcome these losses. We employ the probabilistic roadmap planning algorithm for a robot to reach the ripped mushrooms by avoiding the static and dynamic obstacles in its path. Inverse kinematics (IK) has been employed for letting the hand reach the exact location of a ripped bud and letting the fingers decide a configuration that holds the mushroom. The hand holding the mushroom is then moved vertically up. The action results in mushroom-plucking. First of all, the mushroom cultivation considered here is as an intercrop and is considered to be carried out in discrete clusters on a big farm. To the best of our knowledge, this situation hasn't been considered by any contemporary researchers in their studies.

The requirement for a dynamic environment is for the same. There is no one best algorithm for handling the dynamic environment. So, based upon the parameter values, the probabilistic motion planning has been recommended. IK is not a new idea. However, we didn't find people using it for harvesting. So the combination is a novelty. The objective of this research is to theoretically analyze several possibilities and suggest a feasible framework for a problem of commercial importance. Unlike the other contemporary works in this domain, our proposal doesn't call for any human intervention; the analytical treatment explained in this paper is self-explanatory; it doesn't call for proof by implementation.

The use of a probability road map (PRM) for farm navigation is proposed for robotic mushroom harvesting in a random field. (PRM is a sampling-based 2-step method that includes roadmap construction and querying.) IK is employed for plucking the ripened mushrooms.

In this paper, we briefly sketch the steps of the procedure for farming mushrooms, followed by a concise literature survey of the automation of robotic mushroom harvesting.

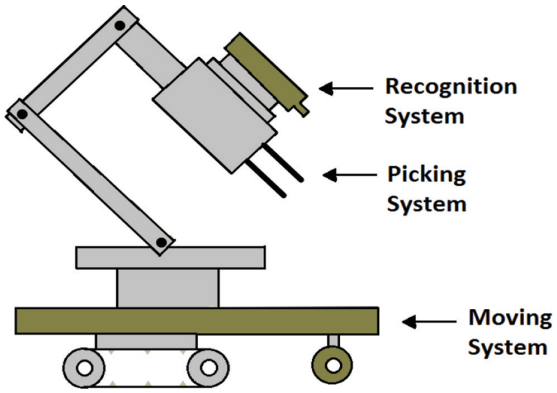

FIGURE 1 | Schematic of a harvesting robot.

Extending the earlier work stated in the survey, we discuss PRM for the planning of robotic motion within the mushroom maze or random plantation, with static obstacles. The method is extended to find the roadmap in environments with dynamic obstacles. It checks whether or not a robot is in an obstacle-free configuration and proceeds accordingly. The method is capable of dealing with robots with many degrees of freedom and having diverse constraints, and it has been shown to be probabilistically complete, i.e., the probability of failure for a planner to find a solution trajectory, if one exists, converges rapidly to zero as the number of collision-free samplings of the workspace increases (1–5). The core of our mushroom harvesting robot is an algorithm for encountering static obstacles by a path-finding robot (6–8). The PRM computes a collision-free path between two ripened mushrooms with a *local planner*.

The basic idea is to check if the roadmap constructed to avoid static obstacles also works with dynamic obstacles, i.e., obstacles moving at a given instant. If it works, then the path is built; if not, the edges that meet the moving obstacles are marked as blocked and construction of an alternative path is attempted (1, 2). A five-step procedure for the PRM in such an environment has been described later.

The design of a dexterous robot hand is driven by the task of plucking the targeted mushroom assigned to it. We propose a two-step process: first, an assembly of two fingers that is analogous to a thumb and a pointing finger of a human hand to get a grip on the stem of the mushroom bud that is to be plucked; in the next step, the stem is uprooted. The joint angles of fingers are calculated by employing IK.

The advantages of the proposed methods are as follows:

(i) Identification of ripened mushrooms by employing image processing

(ii) The path with the least possible collision possibilities by using dynamic PRM gives a more realistic perspective of the environment.

(iii) Minimizing mushroom damage while plucking

(iv) Cost optimization in terms of labor, resources, and plucking cycles

(v) Minimizing the mushroom wastage due to environmental factors like humidity and temperature

The six steps involved in mushroom farming are spawn production, compost preparation, spawning, spawn running, casing, and fruiting. Since these steps are not relevant for motion planning, we will not consider them here.

A mushroom harvesting robot (**Figure 1**) is made up of three components: (i) a recognition system that recognizes mature mushrooms and confirms their positions; (ii) a wheeled moving system that goes along the routes to reach the mature mushrooms; and (iii) a picking system that grasps and plucks the mushrooms at the specified place (9, 10). This study introduces and builds a unique robotic model to conduct moving and selecting actions effectively using input from a recognition system.

After the literature survey, the third section describes the PRM algorithm and the IK model. The details of our proposed robotic hand are provided in the fourth and fifth sections, followed by a discussion in the sixth section of the results of this research, and the conclusion in the seventh section.

Brief literature survey

A maze-like structure is conceived for a robot to move through the field to carry out harvesting activities. Given suitable image specifications, e.g., cap size, proximity to other mushrooms, etc., a typical mushroom harvesting system must be able to identify and pick target mushrooms at suitable times (11). Even mild damage to the mushrooms affects the shelf life and selling price. It is a challenge to design a robotic system with the required precision in terms of size and location of the mushrooms, so as to facilitate picking them and carefully placing them into a container, without causing any damage or contamination to them or their neighbors (12).

A computer model of a mushroom farm suitable for a conceived robot was constructed to test the robot's efficiency. An implementation of the robot was successful in plucking 80% of the mushrooms on a real farm of the same specifications (11). A camera-captured image of a mushroom farm is processed to identify if some of the mushrooms in a group or block are ripe and hence ready for plucking. These mushrooms are picked while keeping the damage to the others at a minimum (13).

Probabilistic road map for motion planning of a robot within a random field

First, we discuss the PRM method for the planning of robotic motion with static obstacles. Then the logic is extended to find a roadmap with dynamic obstacles.

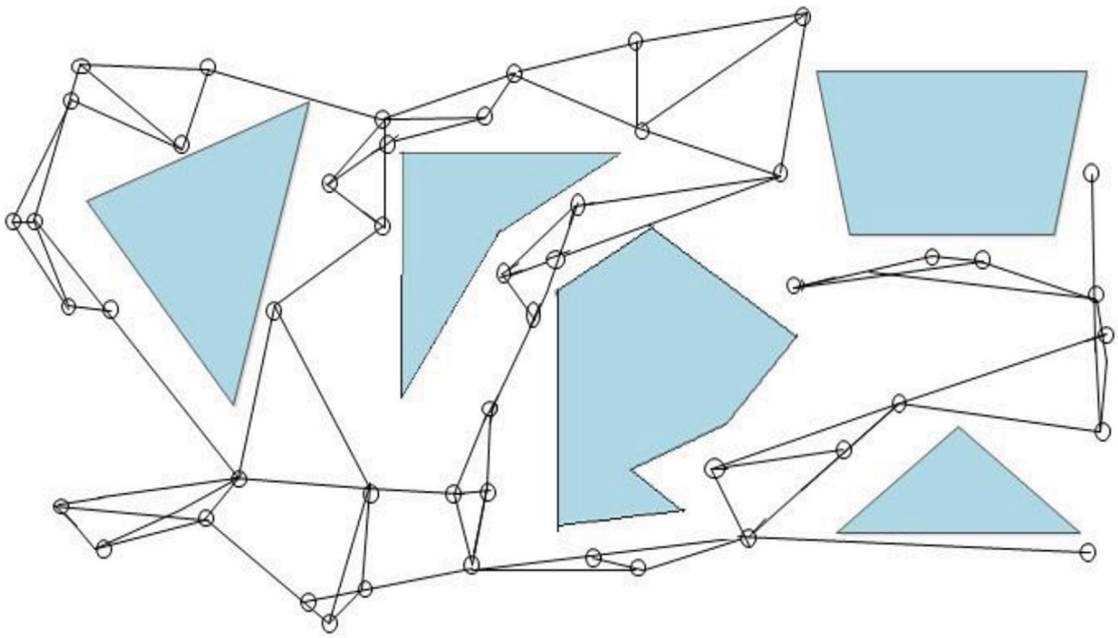

FIGURE 2 | Example of a roadmap for a point robot in two-dimensional Euclidean space. Shaded areas are obstacles. The small circles are nodes of a graph, and the edges represent obstacle-free paths between adjacent nodes.

The PRM is a sampling-based 2-step iterative method that includes roadmap construction and querying (see algorithm and **Figures 2**, **3** later). It determines whether a robot is in an obstacle-free configuration space Q_{free} (for more information, see **Figures 2**, **3**).

The core of our mushroom harvesting robot is a PRM algorithm for encountering static obstacles (2, 7, 8). The following algorithm for the navigation of a robot through a mushroom farm is an implementation of Refs. (14–19).

Algorithm for static obstacles

Definitions

1. A roadmap is an undirected graph G = (V, E), where the nodes in V represent a set of ripened mushrooms, and each edge in E is a collision-free path between two nodes computed by a *local planner*. See **Figure 2**.

2. Nodes q_{init} and q_{goal} are user-provided inputs. They are, respectively, the initial and final nodes in a path to be discovered by the algorithm.

3. Querying: Let Connect Qinit be a list of neighboring nodes in the roadmap in the order of their distances from q_{init}, and similarly, let Connect Q goal be a list of neighboring nodes in the same roadmap in the order of their distances from q_{goal}. Try connecting q_{init} to each of its neighboring nodes, and q_{goal} to its neighboring nodes; call the nodes a' and a'', respectively. Search the graph G for a sequence of edges in E connecting a' to a''. Convert this sequence into a feasible path for the robot by computing the corresponding local paths and concatenating them. The local paths can be stored in the roadmap. The whole sequence, q_{int}-a'-. . .-a''–q_{goal} is a feasible path for a robot. Among the feasible paths, find the shortest path on the roadmap between q_{init} and q_{goal} by employing an appropriate algorithm—one of the A*, D, or D*Lite algorithms (20–24).

Algorithm for static obstacles. Repeat steps S1 and S2 below, until all mushrooms in the node set G are covered

S1 (construction). For a given workspace, construct a roadmap in a probabilistic manner, i.e., randomly select a configuration of nodes (provided by image processing), using some sampling distribution.

S2 (querying). Given an initial configuration q_{init} and goal configuration q_{goal}, find the shortest path connecting q_{init} and q_{goal}.

Remark: The robot is supposed to move and pluck all the mushrooms along this path, removing them from the graph. Then, the two steps of the algorithm are to be repeated.

Figures 2, **3** illustrate the two phases of the iterative path finding algorithm. In **Figure 3**, the shortest path from q_{init} to q_{goal} is marked with thick lines.

Algorithm for dynamic obstacles

The basic idea is to check if the roadmap constructed to avoid static obstacles also works despite dynamically moving obstacles at a given instant. If it works, then the path is built. Otherwise, the edges that meet the moving obstacles are marked as blocked and construction of an alternative path is attempted (25). The two ends of the blocked edges are joined locally using a rapidly exploring random tree (RRT)

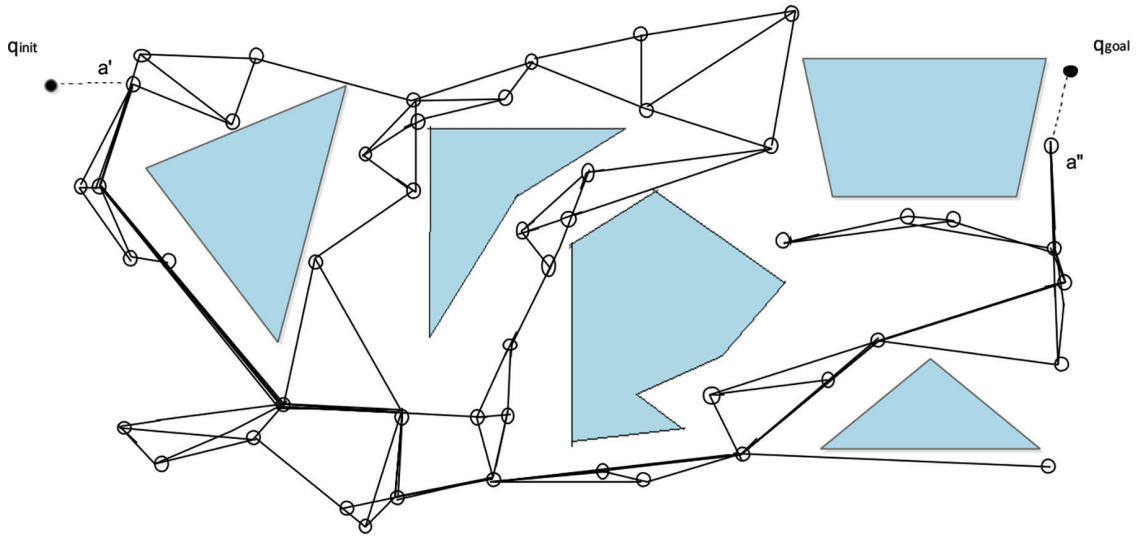

FIGURE 3 | Example of a query with the roadmap. Nodes q_{init} and q_{goal} are first connected to the existing roadmap through nodes a′ and a″. The search algorithm returns the shortest path, denoted by a thick dark line.

technique. RRT is an algorithm that searches non-convex, high-dimensional spaces quickly by randomly constructing a space-filling tree. The tree is built progressively from random samples selected from the search space and is intrinsically inclined to expand toward huge unexplored sections of the issue. It is commonly used in dynamic robotic motion planning (19, 26–28). In a dynamic environment, the initial and goal configurations are also moving entities, and therefore the new path has to be constructed by considering their new positions.

We now describe a five-step procedure for the PRM in dynamic environments.

1. Roadmap labeling and solution path search

After receiving the dynamic updates, the planner iteratively performs the following two operations alternately until a valid path is found or all connections are attempted:

(a) connection of a query node (q_{init} or q_{goal}) to the nodes of the roadmap, and (b) search for a valid path inside the roadmap.

Given that the obstacles are moving, a solution is achieved when all edges stay collision-free. The collision-causing edges are blocked, and the result is used to recreate that specific segment of the roadmap's dynamic connectivity. The modified connectivity is then used in the following iteration to choose the best candidate nodes in the roadmap to locate further probable connections between the query nodes.

2. Query node connections

At each iteration, the candidate nodes are selected by decomposing each statically connected component into three subcomponents: (i) nodes potentially reachable from q_{init} (i.e., there exists a path with no blocked edges); (ii) nodes potentially reachable from q_{goal}; and (iii) nodes that are currently not reachable from the query nodes q_{init} and q_{goal}. This method avoids many costly updates of edges that are not required to answer the connectivity query.

3. Local reconnections

The details of connectivity trails in the roadmap from the q_{init} and q_{goal} are maintained in a rapidly exploring random tree (RRT) structure (27), using which the alternate paths are constructed to connect the end vertices of the blocked edges. A new connection from the query nodes to the static roadmap is attempted when the blocked edge reconnection fails.

4. Node insertion and cycle creation (28)

If the roadmap is unable to provide collision-free connectivity between q_{init} and q_{goal}, a few nodes are added to it and the necessary labeling is performed. The disjoint components in the static roadmap are linked through the connectivity between the new nodes. This technique achieves efficiency by avoiding the creation of unnecessary edges and nodes inside the roadmap.

The four steps discussed earlier are illustrated in **Figure 4**.

5. Edge labeling

The current positions of all the moving obstacles are checked against the edges. An edge is blocked if one or many obstacles are found colliding with it. The label employs the dynamic programming approach. The location information and the results of collision tests are simultaneously stored in an edge-tree structure (**Figure 5**). During the next call to the edge labeling, if the location M_i of a moving object matches a stored location, further computation is not necessary: the results of the previous collision tests are retrieved and further collision tests are cancelled. Otherwise, the new location M_i is inserted into the edge-tree structure while maintaining the details of the last-checked positions. Older positions are removed from the data structure to reduce the size of the roadmap. When an obstacle stops moving, it is treated as a static one.

(Indices for edge, moving obstacle, and key position are for the storage structure).

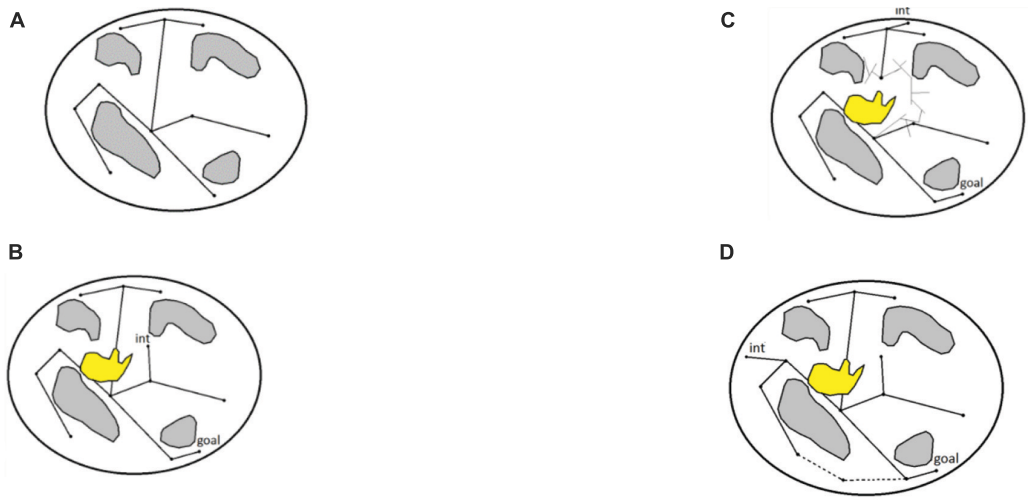

FIGURE 4 | (A) A static roadmap is completed in the configuration space, **(B)** During queries, it is found that a link in a solution path is broken due to a dynamic obstacle, **(C)** An RRT technique is used to reconnect edges broken by dynamic obstacles, and **(D)** if the existing roadmap does not yield a solution, new nodes and edges (dotted lines) are inserted, and thus an obstacle-free path from q_{init} to q_{goal} is found. (Observe that the query nodes q_{init} and q_{goal} are themselves changing dynamically).

Kinematics for robotic hand motion

Getting a roadmap ready is a task that enables a robot to reach the ripened mushrooms at the nearest possible place from its current location. The next task is to model the motion of the robot hand (the end effector) to reach a ripened mushroom.

The design of a dexterous robot hand is driven by the task assigned to it. Different models have been discussed (29, 30). We discuss an assembly of two fingers (analogous to a thumb and a pointing finger of a human hand) that gets a grip on the stem of the mushroom bud to be plucked, and next, the process of uprooting the stem. (A five-finger hand, like that of a human, would be too heavy and complicated for the current purpose, taking up more space and potentially harming neighboring buds).

A diagram of this proposed mushroom plucking robot hand is shown in **Figure 6**. The first finger link in the structure is known as the *base,* and the end link is known as the *end effector*.

The required angular displacements in the finger links through the motions at the finger joints are computed by employing kinematics, i.e., the study of the motion of bodies without consideration of the forces that cause the motion.

Kinematics is of two types: forward and inverse. In the case of the robotic hand, forward kinematics generates the location of the end effector given the angular displacement at each joint. On the contrary, if the location of an end effector is known, IK computes the required angular displacement at each finger joint (31–35).

In the following paragraphs we discuss the design of a robotic hand for automatic mushroom plucking.

The IK computations for the finger simulating the pointing finger are shown later. The coordinates of a mushroom to be plucked are the driving parameters. The computation of the thumb follows the same logic. The thumb differs from the index finger in that it has one fewer joint.

The links have an ordered structure in which each link has its own coordinate system and is positioned relative to the coordinate system of the previous link. The position of the link i in the coordinate system of its ancestor is obtained by computing the joint angle (34, 36–43).

The transformation matrix between two adjacent connecting joints is calculated using the DH parameters in Formula (1) and **Table 1**, where s_i indicates sin θ_i, c_i indicates cos θ_i ($i = 1, 2, 3$), α_{i-1} is the twist angle, a_{i-1} is the length of linkages, and d_i is the offset of the linkages. The transformation matrix of the manipulator finger is obtained by multiplying the transformation matrix of each connecting link ${}_{i}^{i-1}T$ ($i = 1, 2, 3, 4$), which is a function of the three joint variables ($\theta_1, \theta_2, \theta_3, \theta_4$). Where $\theta_4 = 0$.

Step 1: Holding a mushroom stem

(X_i, Y_i, Z_i) represents an axial frame of reference. For $i = 0$, it represents the coordinates of the base; the value of i increases by one to denote the coordinates of the next joint. The link after the last joint is the tip, i.e., the end effector. Hence, (X_3, Y_3, Z_3) is the frame of reference for the end effector of a pointing finger (**Figure 7**), while for the thumb it is (X_2, Y_2, Z_2).

Step 1.1: Compute the D–H parameters of the pointing finger, as in **Figure 8**.

Explanation: The transformation matrix between two adjacent connecting joints can be calculated by the D–H parameters in Formula (1) and **Table 1**. where s_i indicates sin θ_i, c_i indicates cos θ_i ($i = 1, 2, 3$), α_{i-1} is the twist angle, a_{i-1} is the length of linkages, and d_i is the offset of linkages The transformation matrix of the manipulator finger can be obtained by multiplying continuously the transformation matrix of each connecting link ${}_{i}^{i-1}T(i = 1, 2, 3, 4)$, which is

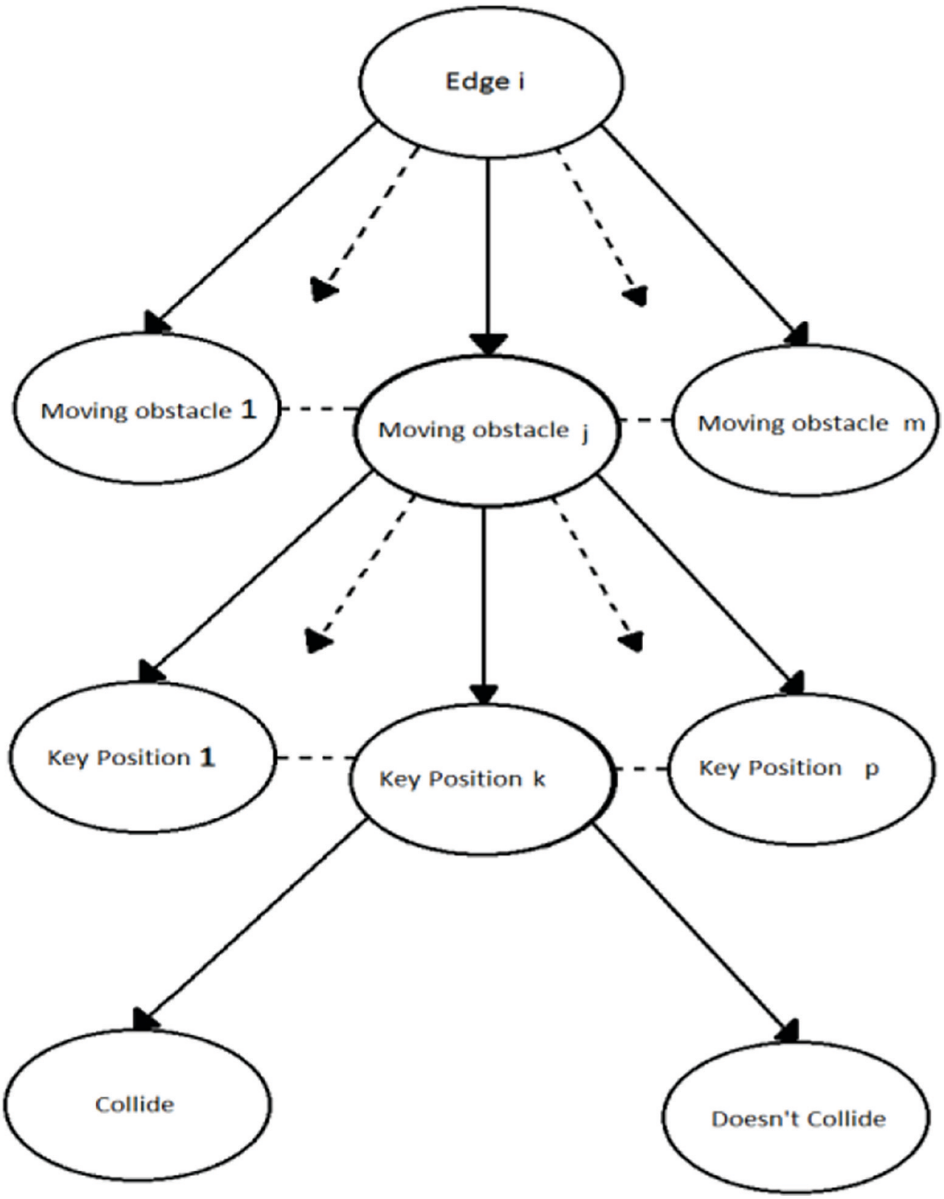

FIGURE 5 | Edge-tree structure.

a function of the three joint variables (θ1, θ2, θ3, θ4), where $\theta^4 = 0$.

Step 1.2: Given the D–H values and the base coordinates of a rigid body, we compute the coordinates of its tip by generating the D–H matrices at all joints [matrices (2), (3), (4), and (5) below] and taking their product (${}^0_4T = {}^0_1T^1_2T^2_3T^3_4T$). Note that $i-1$ is the base of the link and i is the successor link.

$$ {}^{i-1}_{i}T = \begin{bmatrix} C\theta_i & -S\theta_i & 0 & a_{i-1} \\ S\theta_i C\alpha_{i-1} & C\theta_i C\alpha_{i-1} & -S_{\alpha_{i-1}} & -S_{\alpha_{I-1}} d_i \\ C\theta_i S\alpha_{i-1} & C\theta_i S\alpha_{i-1} & C\alpha_{I-1} & C\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1) $$

where S and C represent the sine and cosine functions. Hence, we have

$$ {}^0_1T = \begin{bmatrix} C\theta_1 & -S\theta_1 & 0 & 0 \\ S\theta_1 & C\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2) $$

$$ {}^1_2T = \begin{bmatrix} C\theta_2 & -S\theta_2 & 0 & l_1 \\ S\theta_2 & C\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3) $$

$$ {}^2_3T = \begin{bmatrix} C\theta_3 & -S\theta_3 & 0 & l_2 \\ S\theta_3 & C\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4) $$

$$ {}^{3}_{4}T = \begin{bmatrix} 1 & 0 & 0 & l_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5} $$

$$ {}^{0}_{4}T = \begin{bmatrix} C(\theta_1+\theta_2+\theta_3) & -S(\theta_1+\theta_2+\theta_3) & 0 \\ S(\theta_1+\theta_2+\theta_3) & C(\theta_1+\theta_2+\theta_3) & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} l_1C\theta_1 + l_{2C}(\theta_1+\theta_2) + l_{3C}(\theta_1+\theta_2+\theta_3) \\ l_1S\theta_1 + l_{2S}(\theta_1+\theta_2) + l_{3S}(\theta_1+\theta_2+\theta_3) \\ 0 \\ 1 \end{bmatrix} \tag{6} $$

Inverse kinematics has multiple solutions for a specific position and orientation of the fingertip.

We solve the matrix for a given position p_x, p_y by assuming the following orientation:

$$ {}^{n}_{i}T = \begin{bmatrix} C(\theta_1+\theta_2+\theta_3) & -S(\theta_1+\theta_2+\theta_3) & 0 & p_x \\ S(\theta_1+\theta_2+\theta_3) & C(\theta_1+\theta_2+\theta_3) & 0 & p_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7} $$

Step 1.3: Compute θ

We want to compute θ_1, θ_2, and θ_3 only; we do not need to compute ${}^{4}_{0}T$. We can work with ${}^{3}_{0}T$ as follows:

$$ {}^{3}_{O}T = {}^{0}_{1}T^{1}_{2}T^{2}_{3}T = \begin{bmatrix} C(\theta_1+\theta_2+\theta_3) & -S(\theta_1+\theta_2+\theta_3) \\ S(\theta_1+\theta_2+\theta_3) & C(\theta_1+\theta_2+\theta_3) \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & l_1C\theta_1 + l_2C(\theta_1+\theta_2) \\ 0 & l_1S\theta_1 + l_2S(\theta_1+\theta_2) \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{8} $$

Comparing (7) and (8), we obtain values for p_x and p_y

$$ p_x = l_1C\theta_1 + l_2C(\theta_1+\theta_2) \tag{9} $$

$$ p_y = l_1S\theta_1 + l_2S(\theta_1+\theta_2) \tag{10} $$

Square both sides of the equations (9) and (10), add them and set $C\theta_1{}^2 + S\theta_1{}^2 = 1$

$$ p_x{}^2 + p_y{}^2 = l_1{}^2 + l_2{}^2 + 2l_1l_2C\theta_2 $$

$$ C\theta_2 = \frac{p_x{}^2 + p_y{}^2 - l_1{}^2 - l_2{}^2}{2l_1l_2} \tag{11} $$

$$ S\theta_2 = \sqrt{1 - C\theta_2{}^2} \tag{12} $$

$$ \theta_2 = a\tan(\theta_2, C\theta_2) $$

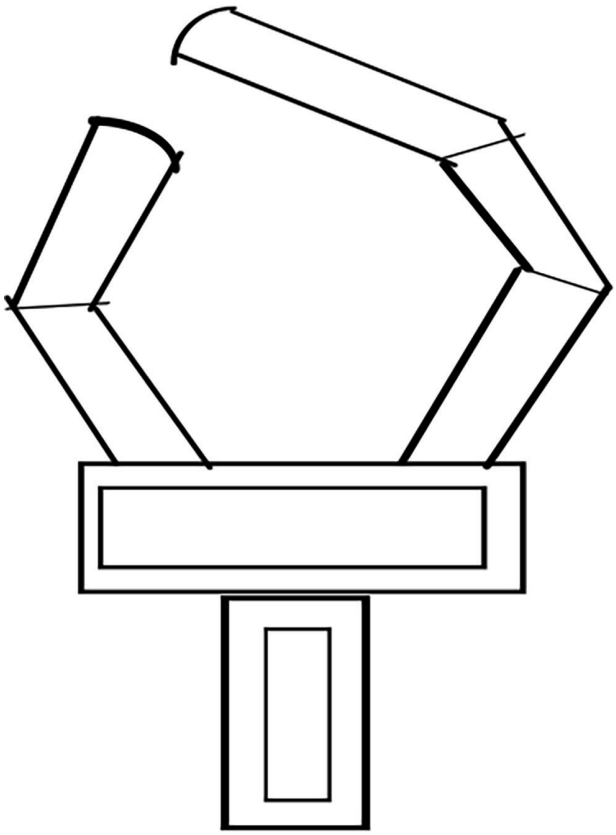

FIGURE 6 | Model of a two-finger robot hand.

Writing equations (9) and (10) in the form

$$ p_x = k_1C\theta_1 + k_2S\theta_1 \tag{13} $$

$$ p_y = k_1S\theta_1 + k_2C\theta_1 \tag{14} $$

where $k_1 = l_1 + l_2C\theta_2$ and $k_2 = l_2S\theta_2$, and assuming $r = \sqrt{k_1{}^2 + k_2{}^2}$, $\gamma = \text{atan}(k_2, k_1)$, $k_1 = rC\gamma$ and $k_2 = rS\gamma$

We substitute these values in (13) and (14), to obtain

$$ \frac{p_x}{r} = C\gamma C\theta_1 + S\gamma S\theta_1 = C(\gamma+\theta_1) $$

$$ \frac{p_y}{r} = C\gamma S\theta_1 + S\gamma C\theta_1 = S(\gamma+\theta_1) $$

$$ \gamma + \theta_1 = atan(p_y/r, p_x/r) = atan(p_y, p_x) $$

$$ \theta_1 = atan(y, x) - atan(k_2, k_1) $$

θ_3 can be obtained by using $C(\theta_1+\theta_2+\theta_3)$ and $S(\theta_1+\theta_2+\theta_3)$

Let $(\theta_1+\theta_2+\theta_3) = \text{atan}(S_\omega, C_\omega) = \omega$

$\theta_3 = atan(S_\omega, C_\omega) - (\theta_1+\theta_2)$,

where $S_\omega = S(\theta_1+\theta_2+\theta_3)$ and $C_\omega = C(\theta_1+\theta_2+\theta_3)$

TABLE 1 | D–H parameters of pointing finger [from Ref. (35)].

# Joint *i*	d_i (joint distance)	a_{i-1} (link length)	α_{i-1} (link twist)	θ_i (joint angle)
1	0	0	0	θ_1
2	0	l_1	0	θ_2
3	0	l_2	0	θ_3
4	0	l_3	0	0

Hence, the joint angles are

$$\theta_1 = atan(p_y, p_x) - atan(k_2, k_1) \quad (15)$$

$$\theta_2 = a\tan(\theta_2, C\theta_2) \quad (16)$$

$$\theta_3 = atan(S_\omega, C_\omega) - (\theta_1 + \theta_2) \quad (17)$$

Step 2: Uprooting the stem of a mushroom

Let the upward force to realize the plucking/uprooting be provided by means of an upward movement of the hand, causing δ change in its y-coordinate; there will be no change in the x-coordinate of the end effector's position nor in the angles at the finger assembly.

Initially, the angle φ at the joint that connects the assembly of two fingers to the wrist-like structure will have some value. The elevation in the end effector's position modifies this angle to the one shown in **Figure 9**.

The following are the inverse kinematic computations that allow the robot-hand to reach its new position ():

$\sin\varphi = \frac{p_y+\delta}{l_0}$, $\cos\varphi = \frac{p_x}{l_0}$ where l_0 is the length of the joint.

$$\tan\varphi = \frac{\sin\varphi}{\cos\varphi} = \frac{p_y + \delta}{p_x}$$

$$\varphi = atan(p_y + \delta, p_x)$$

Observe that steps i and i+1 are carried out at time instances i and i+1, respectively. These activities on the time axis facilitate the preservation of the states of the subassemblies while planning the movements at the joints connecting them. In this case, the position of the fingers holding the stem remains unchanged in the next step while the stem is uprooted.

Algorithm: Formalization of automatic mushroom harvesting robot

Construction of a collision-free path in a farm of ripened mushrooms at some time instant

1. //Data:
2. A = ℓ × d /* the dimension of a rectangular maze formed by random cultivation of mushrooms that provides a specific path for a robot to navigate for plucking mushrooms.

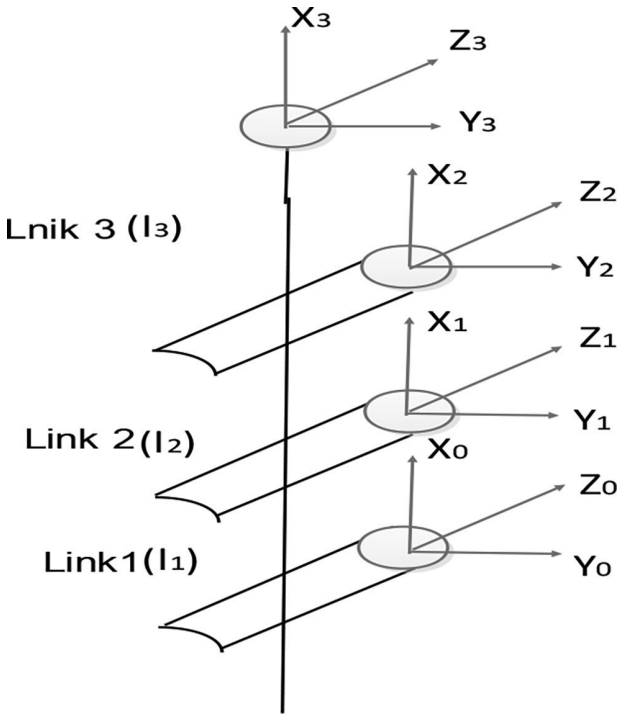

FIGURE 7 | Model of pointing finger.

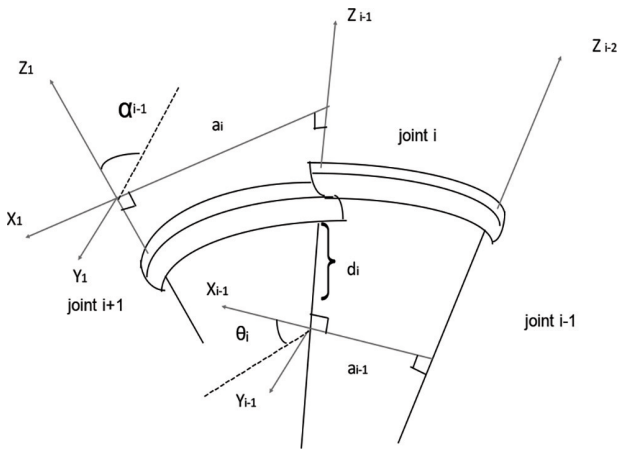

FIGURE 8 | Denavit–Hartenberg (DH) frame for joints.

3. */
4. Let $c_1, c_2, c_3, \ldots, c_n$ cameras in the maze at time instant t_k
5. $\{c_i\}$, $1 \le i \le n$, /* images captured by camera ci, $1 \le i \le n$, */
6. $V = \bigcup_{i=1}^{n} \{c_i\}$ /* the vertex set V for a mushroom plucker to work with.*/
7. //Results:
8. // The average cost of mushroom plucking in this method
9. //Begin
10. G = resultant subgraph from V

11. /* Employ the construction phase of the PRM to find out vertex clusters within radius u of the robot such that there exists a collision free navigation path connecting any two vertices of a cluster. The inter-cluster connectivity is taken care of by a local planner. Let G be the resultant subgraph */
12. W = cost matrix
13. wi j = cost of traversal /* the cost of traversal from the i-th pluckable
14. Mushroom to the j-th pluckable mushroom */
15. Let P_{goal} = { } /* initially. */
16. Let v_s and v_t be two randomly selected vertices of G
17. Employ PRM querying with v_s and v_t on G and get a path P connecting the two. Let V_p be the set of vertices that comprise P.
18. if P is the spanning path:
19. then append P to P_{goal} and stop
20. else:
21. G = G\V_p < G is obtained by deleting the vertices of V_p from the original G >
22. Select v_1 and v_2 randomly from the vertex set of G
23. if $w_{t1} > w_{t2}$:
24. then $v_s = v_2$ and $v_t = v_1$
25. else:
26. $v_s = v_1$ and $v_t = v_2$
27. append vs. to Pgoal
28. Go to step 17
29. Compute the total cost of P_{goal}, i.e., the summation of the costs of traversal from i-th vertex to i+1-th in P_{goal}, where i = 1 to k < k+1 is the length of the path P_{goal} >
30. /* Reaching, gripping and plucking the mushrooms on a given path */
31. /* Let v_1, v_2, ...,v_N be the vertices (of the pluckable mushrooms) on P_{goal}. */

    for i = 1 to N:

    Traverse v_i

    Read the elevation of the pluckable mushroom so that we get its

    coordinates in a three-dimensional frame

    Employ IK to compute the angles at each joint of the robot-hand such that the robot-fingers will reach and hold the pluckable mushroom

    Employ a suitable mechanical process to realize the gripping of the mushroom

    Compute the change in elevation of the mushroom holding hand such that the mushroom will be uprooted

    Employ a suitable process to move the robotic hand and pluck the mushroom.

The average cost of mushroom plucking in this method = Average cost of traversal of a node in path P goal + cost of gripping a mushroom + cost of uprooting the mushroom.

32. //End.

Discussion

Below we have presented algorithms for the automation of mushroom harvesting, with a focus on the following areas:

(i) Identification of ripened mushroom by employing image processing

(ii) Using dynamic PRM to obtain a collision-free path (this is a novel feature not found in the literature)

(iii) No human intervention is required for obstacle avoidance, as this is done by the PRM algorithm itself

(iv) Application of IK to compute the coordinates of a mushroom plucking robot

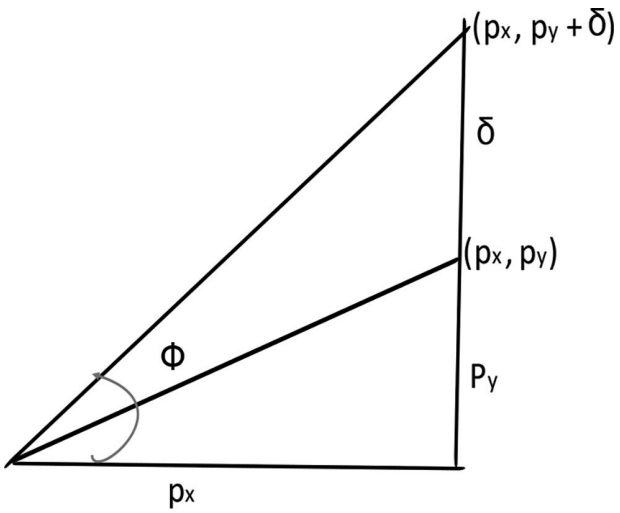

FIGURE 9 | Wrist movement for uprooting mushroom.

TABLE 2 | Approximate costs of manual mushroom harvesting on a 1 acre farm.

Recurring costs per harvesting cycle	
Items	**Cost (Rs)**
Labor	50,000
Fertilizer and spores	20,000
Electricity and water	30,000
Transportation	20,000
Miscellaneous expenses	5,000
Total per harvesting cycle	1,25,000

(v) Minimizing damage to mushrooms while plucking

(vi) Cost optimization in terms of labor, resources, and plucking cycles

(vi) Minimizing mushroom wastage due to environmental factors like humidity and temperature

Brief cost details

Mechanization of operations has arisen in practically every area of the farming sector. This is a side effect of the reduction in the population that engages in farm labor as industrial development takes place. In other words, machines step in whenever and wherever human labor becomes unavailable, inefficient, or expensive.

The all-too-real problem faced by farmers generally—mushroom farmers in the present case—is the non-availability of labor for the timely harvesting of crops. Due to this, mushrooms get over-ripe or damaged, resulting in losses to farmers.

It is stipulated that the problem can be solved by robotic automation. The speed of harvesting is greater, so the duration of harvesting is less. Additionally, machines mitigate human factors like fatigue and lack of diligence. As a result, large-scale and low-cost farming becomes feasible with such automation.

The costs in mushroom farming are listed in **Table 2**.

These figures can vary depending on the circumstances, but the order of magnitude is correct. If three crops can be obtained yearly, the annual profit will be in the range of Rs. 6,00,000.

Note: It is important to account for the one-time capital expense of Rs. 3,00,000 toward the raw material for constructing the sunshade and the additional fabrication cost. In addition, there is a periodic maintenance cost. But we have not done so, for it would take us far away from our research. It is suffice to note that these costs can be amortized over time and covered by the substantial profits.

It is altogether another matter to estimate the cost of designing and fabricating a robotic hand, mounted on a chassis that can travel on a farm. This too falls well outside the scope of this research. We assume that as international products reach our country and as indigenous R&D activities take place, such a robot will become available in the future.

Summary and conclusion

Diverse technologies have been employed in the automation of agricultural activities. Mushroom harvesting is a step-by-step procedure, so it is possible to design a harvesting robot made out of three major parts: a recognition system, a moving system, and a plucking system.

The recognition system gets photographs of the mushrooms from several cameras mounted in the field. Image processing algorithms are employed to identify ripened mushrooms. This software converts the field into a graph, with individual ripened mushrooms as vertices, and with edges joining vertices corresponding to neighboring mushrooms. (Details of the image processing algorithms are outside the scope of the present study.) This graph will be altered and traversed by the PRM algorithms so as to avoid static or dynamic obstacles. The moving system is a wheeled chassis controlled by the PRM algorithm. The plucking system is a robotic hand. Angles at the different joints of the dexterous robotic fingers are computed by employing IK such that first the fingertips reach the ripened mushroom bud and hold it; next the fingers move upward so that the mushroom is plucked; and third, the robotic hand places the mushroom in a container without damaging it.

The novelty of our research is as follows:

(i) This is the first time that PRM algorithms are being proposed for navigation inside mushroom farms.

(ii) Unlike previous research in mushroom harvesting, mushrooms are not planted in a grid or some pattern but are randomly distributed.

(iii) No human intervention is required at any stage of harvesting.

(iv) Robotic automation reduces crop wastage due to unavailability of labor and untimely harvesting of mushrooms.

(v) Harvesting and other expenses are reduced, compared to those with human labor.

(vi) A kinematic model of a two-finger dexterous hand with three degrees of freedom for plucking mushrooms was developed using the Denavit–Hartenberg method.

(vii) IK techniques of reaching the ripened mushroom give more precision in plucking.

(viii) There will be no limitation or restriction on large scale cultivation and harvesting, and this will provide economies of scale.

References

1. Svestka P. *Robot Motion Planning Using Probabilistic Roadmaps*, PhD thesis. Utrecht: Utrecht University (1997).
2. Thrun S, Burgard W, Fox D. *Probabilistic Robotics*. Freiburg: University of Freiburg (2005).
3. Svestka P, Overmars MH. Motion Planning for Car-like Robots, a Probabilistic Learning Approach. *Int J Robot Res.* (1997) 16:119–43.
4. Svestka P, Overmars MH. Coordinated Path Planning for Multiple Robots. *Robot Autonom Syst.* (1998) 23:125–52.
5. Wilmarth SA, Amato NM, Stiller PF. MAPRM: A probabilistic Roadmap Planner with Sampling on the Medial Axis of the Free Space. *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*. Detroit, MI (1999).
6. Dale L. *Optimization Techniques for Probabilistic Roadmaps*, PhD thesis. College Station, TX: Texas A&M University (2000).

7. Cortes J, Simeon T, Laumond JP. A Random Loop Generator for Planning the Motions of Closed Kinematic Chains Using PRM Methods. *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*. Washington, DC (2002).
8. Dale L, Amato N. Probabilistic Roadmaps-Putting it All Together. *Proc IEEE Int. Conf. Robot Automat.* (2001) 2001:1940–7.
9. Bachche S. Deliberation on Design Strategies of Automatic Harvesting Systems: A Survey. *Robotics* (2015) 4:194–222. doi: 10.3390/robotics4020194
10. Baur J, Schütz C, Pfaff J, Buschmann T, Ulbrich H. Path Planning for a Fruit Picking Manipulator. *Proceedings International Conference of Agricultural Engineering*. Zurich: (2014).
11. Reed N, Miles SJ, Butler J, Baldwin M, Noble R. Automatic Mushroom Harvester Development. *J Agric Engng Res.* (2001) 78:15–23.
12. Jayaselan H, Ismail W, Ahmad D. Manipulator Automation for Fresh Fruit Bunch (FFB) Harvester. *Int J Agric Biol Eng.* (2012) 5:7.
13. Rowley J. *Developing Flexible Automation for Mushroom Harvesting (Agaricus bisporus)*, Ph.D Thesis. Warwick: The University of Warwick (2009).
14. Ulbrich H, Baur J, Pfaff J, Schuetz C. Design and Realization of a Redundant Modular Multipurpose Agricultural Robot, DINAME 2015. *Proceedings of the XVII International Symposium on Dynamic Problems of Mechanics*. Brazil (2015).
15. Correll N, Arechiga N, Bolger A, Bollini M, Charrow B, Clayton A, et al. Building a Distributed Robot Garden. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. St. Louis (2009).
16. Han L, Amato N. A Kinematics-Based Probabilistic Roadmap Method for Closed Chain Systems. *Proceedings of the Workshop on Algorithmic Foundations of Robotics (WAFR'00)*. Berlin (2000).
17. Hofstra O, Nieuwenhuisen D, Overmars MH. *Improving the Path Quality for Probabilistic Roadmap Planners, to appear*. Northfield, MN: PRM (2002).
18. Barraquand J, Kavraki L, Latombe J-C, Li T-Y, Motwani R, Raghavan PA. Random Sampling Scheme for Path Planning. *Int J Robot Res.* (1997) 16:759–74.
19. Van Den Bergen G. *Collision Detection in Interactive 3D Computer Animation, PhD thesis*. Eindhoven: Eindhoven University (1999).
20. Bohlin R, Kavraki LE. Path Planning Using Lazy PRM. *Proceedings of the IEEE International Conference on Robotics and Automation*. St Paul, MN (2000).
21. Boor V, Overmars MH, Van Der Stappen AF. The Gaussian Sampling Strategy for Probabilistic Roadmap Planners. *Proceedings of the IEEE International Conference on Robotics and Automation*. St Paul, MN (1999).
22. van den Berg J. *Path Planning in Dynamic Environments*, Ph D thesis. Utrecht: Utrecht University (2007).
23. Koenig S, Likhachev M. D*lite. *Proceedings of the IEEE International Conference on Artificial Intelligence*. Edmonton (2002).
24. Stentz A. The Focussed D* Algorithm for Real-Time Replanning. *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. Macao (1995).
25. Choset H, Lynch KM, Hutchinson S, Kantor GA, Burgard W, Kavraki LE, et al. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Cambridge, MA: MIT Press (2005).
26. Mohanan MG, Salgoankar A. A Survey of Robotic Motion Planning in Dynamic Environments. *Robot Autonom Syst* (2018) 100:171–85.
27. Van Den Berg JP, Overmars MH. Roadmap-Based Motion Planning in Dynamic Environments. *IEEE Trans Robot.* (2005) 21:885–97.
28. LaValle SM. Rapidly-exploring random trees: A new tool for path planning. *Proceedings of the IEEE International Conference on Robots and Systems*. Ames, IA (2000).
29. Kuffner J, LaValle S. RRT-Connect: An efficient Approach toSingle Query Path Planning. *Proceedings of the IEEE International Conference on Robotics and Automation*. Sendai: (2000).
30. Jaillet L, Siméon T. A PRM-Based Motion Planner for Dynamically Changing Environments. *Proceedings of the IEEE International Conference on Robots & Systems*. Sendai (2004).
31. Kucuk S, Bingul Z. *Robot Kinematics: Forward and Inverse Kinematics, Industrial Robotics: Theory, Modelling and Control*. Jakarta Selatan: InTech (2006).
32. Guo J, Jiangnan N. Analysis and Simulation on the Kinematics of Robot Dexterous Hand. *Proceedings of the International Conference on Electronics, Network and Computer Engineering (ICENCE 2016)*. Yinchuan (2006).
33. Bertram D, Kuffner J, Dillmann R, Asfour T. An Integrated Approach to Inverse Kinematics and Path Planning for Redundant Manipulators. *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*. Orlando, FL (2006).
34. Chan KW. *Closed- form and Generalised Inverse Kinematic Solutions for Animating the Human Articulated Structure*. Perth: Curtin University of Technology (1996).
35. Craig JJ. *Introduction to Robotics: Mechanics and Control*. Boston, MA: Addison- Wesley (1994).
36. Novakovic ZR, Nemee B. A Solution of the Inverse Kinematics Problem Using the Sliding Mode. *Proceedings of the IEEE Transactions on Robotics and Automation*. New Orleans, LA (1990). doi: 0.109/70.54740
37. Kuffner J. Effective Sampling and Distance Metrics for 3D Rigid Body Path Planning. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2004)*. New Orleans, LA (2004). doi: 10.1109/ROBOT.2004.1308895
38. Welman C. *Inverse Kinematics and Geometric Constraints for Articulated Figure Manipulation*. Burnaby, BC: Simon Fraser University (1993).
39. Paul RP. *Robot Manipulators: Mathematics, Programming and Control*. Cambridge, MA: MITPress (1981).
40. Watt A, Watt M. *Advanced Animation and Rendering Techniques*. Boston, MA: Addison- Wesley (1992).
41. Barinka L, Berka I. *Inverse Kinematics- Basic Methods*. Prague: Czech Technical University (2002).
42. Mohanan MG, Salgaonkar A. Probabilistic Approach to robot motion planning in dynamic environments. *SN Comput Sci.* (2020) 1:181.
43. Mohanan MG, Salgaonkar A. *Ant colony optimization and Firefly Algorithms for robotic motion planning in dynamic environments. Engineering reports*. Hoboken, NJ: Wiley Publications (2020).