Robotic Mushroom Harvesting by Employing Probabilistic Road Map and Inverse Kinematics

M. G. Mohanan* and Ambuja Salgaonkar

University Department of Computer Science, University of Mumbai, India
*Corresponding author: mgmohanam@gmail.com; mohanan@udcs.mu.ac.in

Abstract. A collision-free path to a destination position in a random farm is determined using a probabilistic roadmap (PRM) 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 (IK) 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 paper provides pointers to the contemporary research on employing robotics in agriculture. The emphasis is on automation of the process of mushroom plucking. The method is applicable for plucking of mushrooms if they are cultivated in a bed or even otherwise where they are cultivated randomly in a farm. Many times it happens that mushrooms are ready but there is hardly any skilled labour available to pluck them; this situation causes a significant loss to the mushroom farmers as they get perished 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 to the ripped mushrooms by avoiding the static and dynamic obstacles in its path. Inverse kinematics has been employed for letting the hand to reach to the exact location of a ripped bud and letting the fingers to decide a configuration that holds the mushroom. The hand holding the mushroom is then moved vertically up, the action results into mushroom-plucking. First of all the mushroom cultivation considered here is as an inter-crop and is considered to be carried out in the discrete clusters in a big farm. To the best of our knowledge this situation hasn’t been considered by any contemporary researchers in their study.

The need of 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. Inverse kinematics 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 analyse several possibilities and suggest a feasible framework for a problem of commercial importance. Unlike the other contemporary work in this domain, our proposal doesn’t call for any human intervention; Analytical treatment explained in this paper is self-explaining that it doesn’t call for a proof by implementation.

Robotic mushroom harvesting in a random field is proposed by employing a probability road map (PRM) for navigation in the farm. (PRM is a sampling based 2-step method that includes roadmap construction and querying.) Inverse kinematics is employed for plucking the ripened mushrooms.

Below we briefly sketch the steps of the procedure for farming mushrooms, followed by a concise literature survey of the automation of robotic mushroom harvesting. Extending the earlier work stated in the survey, we discuss PRM for the planning of a 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 the static obstacles by a path finder 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 the static obstacles also works with dynamic obstacles, i.e., obstacles moving at a given instant. If it works then the path is built else the edges that meet the moving obstacles are marked as blocked and construction of an alternative paths is attempted [1, 2]. A five step procedure for the PRM in such environment has been described below.

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 Inverse kinematics.

Advantages of the proposed methods are as follows:

(i) Identification of ripened mushroom by employing image processing
(ii) Path with minimising collision possibilities by using dynamic PRM gives a more realistic perspective of the environment,
(iii) Minimising mushroom damages while plucking
(iv) Cost optimisation in terms of labour, resources and plucking cycles,
(v) Minimising 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 recognises 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 an unique robotic model to conduct moving and selecting actions effectively using input from a recognition system.

After the literature survey, Section “Probabilistic Road Map for Motion Planning of a Robot Within a Random Field” describes the PRM algorithm and the IK model. The details of our proposed robotic hand are provided in Sections “Kinematics for Robotic Hand Motion” and “Algorithm: Formalization of Automatic Mushroom Harvesting Robot”, followed by a discussion in Section “Discussion” of the results of this research, and the conclusion in the Section “Brief Cost Details”.

BRIEF LITERATURE SURVEY

A maze like structure is conceived for a robot to move through the field for carrying 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 producing any damage or contamination to them or their neighbours [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 in 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 probabilistic road map (PRM) method for the planning of robotic motion with static obstacles. Then the logic is extended to find a roadmap with dynamic obstacles.
Robotic Mushroom Harvesting by Employing Probabilistic Road Map and Inverse Kinematics

PRM is a sampling based 2-step iterative method that includes roadmap construction and querying [see algorithm and Figures 2 and 3 below]. It checks if a robot is in an obstacle free configuration space $Q_{\text{free}}$ [for details see Figures 2 and 3 below].

The core of our mushroom harvesting robot is a PRM algorithm for encountering static obstacles [2, 7, 8]. The following algorithm for navigation of a robot through a mushroom farm is an implementation of [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 below.

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

3. Querying: Let ConnectQinit be a list of neighbouring nodes in the roadmap in the order of their distances from $q_{\text{init}}$, and similarly, let ConnectQgoal be a list of neighbouring nodes in the same roadmap in the order of their distances from $q_{\text{goal}}$. Try connecting $q_{\text{init}}$ to each of its neighbouring nodes, and $q_{\text{goal}}$ to its neighbouring 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, $q_{\text{init}} - a' - \cdots - a'' - q_{\text{goal}}$ is a feasible path for a robot. Among the feasible paths, find the shortest path on the roadmap between $q_{\text{init}}$ and $q_{\text{goal}}$ by employing an appropriate algorithm—one of the A*, D and 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 $q_{\text{init}}$ and goal configuration $q_{\text{goal}}$, find the shortest path connecting $q_{\text{init}}$ and $q_{\text{goal}}$.

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

Figures 2 and 3 below illustrate the two phases of the iterative path finding algorithm. In Figure 3, the shortest path from $q_{\text{init}}$ to $q_{\text{goal}}$ 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) 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 towards huge unexplored sections of the issue. It is commonly utilised in robotic motion planning in dynamic contexts [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.

A five step procedure for the PRM in dynamic environments is described below.

1. Roadmap labelling 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\textsubscript{init} or q\textsubscript{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 utilised 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 of the query nodes.

2. Query node connections

At each iteration, the candidate nodes are selected by decomposing each statically connected component into three sub-components: (i) nodes potentially reachable from q\textsubscript{init} (i.e., there exists a path with no blocked edges), (ii) nodes potentially reachable from q\textsubscript{goal}, and (iii) nodes that are currently not reachable from the query nodes q\textsubscript{init} and q\textsubscript{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\textsubscript{init} and q\textsubscript{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]

In case the roadmap is not able to provide a collision-free connectivity between q\textsubscript{init} and q\textsubscript{goal}, then a few nodes are added to it and the corresponding labelling is carried out. 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 above are illustrated in Figure 4.

5. Edge labelling

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 labelling 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-labelling, if location M\textsubscript{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\textsubscript{i} is inserted in the edge-tree structure, while maintaining the details of the last-checked positions. Older positions are removed from the data structure in order to reduce the size of the roadmap. When an obstacle stops moving, it is treated as a static one.

![Figure 4](image-url)  
**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\textsubscript{init} to q\textsubscript{goal} is found. (Observe that the query nodes q\textsubscript{init} and q\textsubscript{goal} are themselves changing dynamically).

![Figure 5](image-url)  
**Figure 5.** Edge-tree structure (indices for edge, moving obstacle and key position are for the storage structure).
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. Diverse 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. (Arguably, a five-finger hand like that of a human will be a too-heavy and complicated assembly for the present purpose, as it would take more space and may harm the neighbouring 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, inverse kinematics 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 inverse kinematics computations for the finger simulating the pointing finger are shown below. The coordinates of a mushroom to be plucked are the driving parameter. The computation of the thumb follows the same logic. The difference is that the thumb has one less 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 D-H parameters in Formula (1) and the Table 1, where \( s_i \) indicates \( \sin \theta_i \), \( c_i \) indicates \( \cos \theta_i \) \((i = 1, 2, 3, 4)\), \( a_{i-1} \) 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 \( T_{i-1}^i \) \((i = 1, 2, 3, 4)\), which is a function with 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 1 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 the Table 1. where \( s_i \) indicates \( \sin \theta_i \), \( c_i \) indicates \( \cos \theta_i \) \((i = 1, 2, 3)\), \( a_{i-1} \) 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 \( T_{i-1}^i \) \((i = 1, 2, 3, 4)\), which is a function with the three joint variables \((\theta_1, \theta_2, \theta_3, \theta_4)\), 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 \((T = T_4^3 T_3^2 T_2^1 T_1^0)\).

Note that \( i = 1 \) is the base of the link and \( i \) is the successor link.

\[
T_{i-1}^i = \begin{bmatrix}
C\theta_i & -S\theta_i & 0 & a_{i-1} \\
S\theta_i C\alpha_{i-1} & C\theta_i C\alpha_{i-1} & -S\alpha_{i-1} & -S\alpha_{i-1} d_i \\
C\theta_i S\alpha_{i-1} & C\theta_i S\alpha_{i-1} & C\alpha_{i-1} & C\alpha_{i-1} d_i \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

(1)

<table>
<thead>
<tr>
<th>Joint</th>
<th>( d_i )</th>
<th>( a_{i-1} )</th>
<th>( a_{i-1} )</th>
<th>( \theta_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>( \theta_1 )</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>( l_1 )</td>
<td>0</td>
<td>( \theta_2 )</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>( l_2 )</td>
<td>0</td>
<td>( \theta_3 )</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>( l_3 )</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 6. Model of a two-finger robot hand.

Table 1. D-H parameters of pointing finger (From [35]).
where \( S \) and \( C \) represent the sine and cosine functions. Hence we have

\[
0^4_T = \begin{bmatrix}
C(\theta_1 + \theta_2 + \theta_3) & -S(\theta_1 + \theta_2 + \theta_3) & 0 & 0 \\
S(\theta_1 + \theta_2 + \theta_3) & C(\theta_1 + \theta_2 + \theta_3) & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
\frac{1}{2}T = \begin{bmatrix}
C\theta_2 & -S\theta_2 & 0 & l_1 \\
S\theta_2 & C\theta_2 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
\frac{2}{3}T = \begin{bmatrix}
C\theta_3 & -S\theta_3 & 0 & l_2 \\
S\theta_3 & C\theta_3 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
\frac{3}{4}T = \begin{bmatrix}
1 & 0 & 0 & l_3 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
l_1C\theta_1 + l_2C(\theta_1 + \theta_2) + l_3C(\theta_1 + \theta_2 + \theta_3) \\
l_1S\theta_1 + l_2S(\theta_1 + \theta_2) + l_3S(\theta_1 + \theta_2 + \theta_3)
\]

IK 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:

\[
\begin{bmatrix} 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

Step 1.3: Compute \( \theta \)

As we want to compute \( \theta_1, \theta_2 \) and \( \theta_3 \) only, we do not need to compute \( \frac{3}{4}T \), but we can work with \( \frac{1}{2}T \) as follows:

\[
\frac{3}{4}T = \frac{1}{2}T \cdot \frac{2}{3}T = \begin{bmatrix}
C(\theta_1 + \theta_2 + \theta_3) & -S(\theta_1 + \theta_2 + \theta_3) & 0 & p_x \\
S(\theta_1 + \theta_2 + \theta_3) & C(\theta_1 + \theta_2 + \theta_3) & 0 & p_y \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

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

\[
p_x = l_1C\theta_1 + l_2C(\theta_1 + \theta_2) \quad \text{(9)}
\]

\[
p_y = l_1S\theta_1 + l_2S(\theta_1 + \theta_2) \quad \text{(10)}
\]

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

\[
p_x^2 + p_y^2 = l_1^2 + l_2^2 + 2l_1l_2C\theta_2
\]

\[
C\theta_2 = \frac{p_x^2 + p_y^2 - l_1^2 - l_2^2}{2l_1l_2} \quad \text{(11)}
\]

\[
S\theta_2 = \pm \sqrt{1 - C\theta_2^2} \quad \text{(12)}
\]

\[
\theta_2 = \text{atan}(S\theta_2, C\theta_2)
\]

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

\[
p_x = k_1C\theta_1 + k_2S\theta_1 \quad \text{(13)}
\]

\[
p_y = k_1S\theta_1 + k_2C\theta_1 \quad \text{(14)}
\]

where \( k_1 = l_1 + l_2C\theta_2 \) and \( k_2 = l_2S\theta_2 \), and assuming \( r = \sqrt{k_1^2 + k_2^2} \), \( \gamma = \text{atan}(k_2, k_1) \), and \( k_1 = rC\gamma \) and \( k_2 = rS\gamma \).
We substitute these values in (13) and (14), to obtain

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

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

\[
\gamma + \theta_1 = \tan(p_y/r, p_x/r) = \tan(p_y, p_x)
\]

\[
\theta_1 = \tan(y, x) - \tan(k_2, k_1)
\]

\[
\theta_3 = \tan(S\omega, C\omega) - (\theta_1 + \theta_2),
\]

where

\[
S\omega = S(\theta_1 + \theta_2 + \theta_3)
\]

and

\[
C\omega = C(\theta_1 + \theta_2 + \theta_3)
\]

Hence the joint angles are

\[
\theta_1 = \tan(p_y, p_x) - \tan(k_2, k_1)
\]

(15)

\[
\theta_2 = \tan(S\theta_2, C\theta_2)
\]

(16)

\[
\theta_3 = \tan(S\omega, C\omega) - (\theta_1 + \theta_2)
\]

(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 \( \delta \) 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 \( \phi \) 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 inverse kinematic computations letting the robot-hand to its new position \((p_x, p_y + \delta)\) are given below:

\[
\sin \phi = \frac{p_y + \delta}{l_0}, \quad \cos \phi = \frac{p_x}{l_0}
\]

where

\( l_0 \) is the length of the joint.

\[
\tan \phi = \frac{\sin \phi}{\cos \phi} = \frac{p_y + \delta}{p_x}
\]

\[
\phi = \tan(p_y + \delta, p_x)
\]

Observe that step \( i \) and step \( i + 1 \) 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 sub-assemblies 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.

Figure 9. Wrist movement for uprooting mushroom.

**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. \( A = \ell \times d \) /* 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.*/
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 \leq i \leq n, */ images captured by camera \( c_i \),
6. \( V = \cup_{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 \( P \).
if \( P \) is the spanning path:
then append \( P \) to \( P_{goal} \) and stop
else:
\( G = G \setminus V_p < G \) is obtained by deleting the vertices of \( V_p \) from the original \( G \)
Select \( v_1 \) and \( v_2 \) randomly from the vertex set of \( G \)
if \( w_{t1} > w_{t2} \):
then \( v_s = v_2 \) and \( v_t = v_1 \)
else:
\( v_s = v_1 \) and \( v_t = v_2 \)
append \( v_s \) to \( P_{goal} \)
Go to step 17
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 \( v_1, v_2, \ldots, v_N \) 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.

**DISCUSSION**

Below we have presented algorithms for the automation of mushroom harvesting, with the 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,
(v) Minimising damage to mushrooms while plucking,
(vi) Cost optimisation in terms of labour, resources and plucking cycles,
(vi) Minimising 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 reduction in population that engages in farm labour as industrial development takes place. In other words, machines step in whenever and wherever human labour 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 labour for the timely harvesting of crops. Due to this, mushrooms get over-ripe or damaged and result in losses to farmers.

It is stipulated that the problem can be solved by robotic automation. The speed of harvesting is more, 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 as below.

**Table 2. Approximate costs of manual mushroom harvesting on a one acre farm**

<table>
<thead>
<tr>
<th>Item</th>
<th>Cost (Rs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Labour</td>
<td>50,000</td>
</tr>
<tr>
<td>Fertiliser and spores</td>
<td>20,000</td>
</tr>
<tr>
<td>Electricity and water</td>
<td>30,000</td>
</tr>
<tr>
<td>Transportation</td>
<td>20,000</td>
</tr>
<tr>
<td>Miscellaneous expenses</td>
<td>5,000</td>
</tr>
<tr>
<td><strong>Total per harvesting cycle</strong></td>
<td><strong>1,25,000</strong></td>
</tr>
</tbody>
</table>

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 towards 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 will take us far away from our research. It suffices to note that these costs can be amortized over time and covered by the substantial profits.

It is altogether a different 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 CONCLUSIONS

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: recognition system, moving system and 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 joint of the dexterous fingers and hold it; next the fingers move upwards 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 labour and untimely harvesting of mushrooms.
(v) Harvesting and other expenses are reduced, compared to those with human labour.
(vi) Kinematic model of a two-finger dexterous hand with 3 degrees of freedom for plucking mushrooms was developed using the Denavit-Hartenberg method.
(vii) Inverse kinematics techniques of reaching the ripened mushroom give more precision of plucking.
(viii) There will be no limitation or restriction on large scale cultivation and harvesting and this will provide economies of scale.

CONFLICT OF INTEREST

On behalf of all authors, the corresponding author states that there is no conflict of interest.

REFERENCES


[34] Kwan W. Chan, Closed-form and Generalised Inverse Kinematic Solutions for Animating the Human Articulated Structure, Curtin University of Technology, 1996.


[38] Ch. Welman. Inverse Kinematics and Geometric Constraints for Articulated Figure Manipulation, Simon Fraser University, 1993.


[41] Lukas Barinka, Ing. Roman Berka, Inverse Kinematics – Basic Methods, Czech Technical University, Prague.
