Simulation of Virtual Environment for Scout Mobile Robot

MozhganKhoylo1, MoharamHabibNejadKorayem2, VahidAzimirad3

1. Department of Computer-Mechatronic, Science and Research Branch of Islamic Azad University Tehran, IRAN
2. College of Mechanical Engineering, Iran University of Science & Technology Tehran, Iran
3. Biomechatronic Research Lab, School of Engineering Emerging Technologies University of Tabriz

Corresponding Author email: mojgankho@aol.com

ABSTRACT: This study has been set out to fill the perceived gaps within the context of mobile robots literature. For this purpose, a real-life robot which is known as Scout – and one set of which has hitherto been purchased by Iran University of Science and Technology – has been adopted to carry out investigations on the motion-planning problem. The optimal path for the base of the manipulator robot was designed and developed benefitting from the Kinematic and Dynamic functions along with the Path profile; withal, Optimal Control was also utilized before applying Pontryagins minimum principle. Moreover, in contemplation of devising autonomous obstacle avoidance alongside diminishing the risk of being lost or deviated from a target point, over the course of the next stage, so-called Artificial Intelligence has been implemented in the noted robot by means of the Artificial Neural Networks. Thereafter, Scout has been simulated in a three-dimensional environment, whereby a comparison is taken place to contrast the differences resulted from the 2D simulation. This comparison not only verifies the exact compliance of the results of both the 2D and 3D simulations, but also proves the benefits of the 3D simulation, which provides a more realistic analysis of robot’s motion-planning. At length, the findings of this research affirm that enhancing robot with the Kinematic and Dynamic functions, Path profile, Optimal Control and Artificial Neural Network promotes autonomous traits of the robot that facilitates collision-free movements. Furthermore, this study might serve as a test-bed for conducting investigations on the routing of the Scout robot prior to any real-time implementations.

Key Words: Artificial Intelligence; Mobile Robots; Navigation; Neural Networks; Obstacle Avoidance; Simulation.

Abbreviations: nDLCC-Dynamic Load Carrying Capacity; MADL-Maximum Allowable Dynamic Load; MLP-Multilayer Perceptron; NN-Neural Networks; RFID-Radio Frequency Identification.

INTRODUCTION

In recent times, there has been a huge leap toward making intelligent mobile robots. Generally, these mobile robots are designed to mimic performance of various tasks and to replace man-power, due to their inherent resources-savvy characteristics, e.g., of time and/or cost. In order to reduce the errors and to optimize movement of the mobile robots, Kinematics, Dynamics, and Optimal path of the robots are modeled by mathematical functions. Within this context, a research has been conducted by Korayem et al. (2010) to find Maximum Allowable Dynamic Load (MADL) of wheeled robots in presence of obstacle and moving boundary condition; nonetheless, no artificial intelligence has been incorporated throughout this research, hence, the scope of this study would be to accommodate the actual robot – Scout – used in the previous work with artificial intelligence. Further, designing such robots necessitate analyzing and modeling the motion-planning problem. The resolution of the motion-planning problem in robotics can be explained as planning a continuous path from the initial position of the robot to the goal position considering robot’s initial location/orientation, location and orientation of the destination, and a set of obstacles located in a workspace. Motion-planning problem encompasses two sub-problems of ‘Findspace’ and ‘Findpath’. The findspace problem involves construction of the configuration space that contains the robot and the obstacles; whereby, the findpath problem engages preparation of a collision-free path for the robots from a definite start point to the destination. In this study, the Scout robot acquires its environment – Finside problem – utilizing the training data which is obtained from the ultrasonic sensors. The attained data is then adopted by robot to autonomously devise an obstacle-free path to reach the desired destination – Findpath. To justify this unaided trait by robots, it requires feigning decision
making structure. Since Neural Networks are able to learn effectively and that they offer good performance for approximation of sophisticated nonlinear functions, in this study, artificial intelligence is implemented exploiting two Neural Logic Networks (Neulonets – an artificial neural network).

Previous Studies

There have been considerable amounts of research carried out to address autonomous mobile robots. Evans et al. (2010) have proposed a scheme that resembles aggregation spirit prevalent among species. Evans et al. (2010) investigated and modeled the self-assembly of miniaturized robots. In favor of this purpose, they have used a swarm of Alice robots. Specifically they have studied formation of chains of a desired length respecting two states, a deterministic controller for the first state, and for the second state, a probabilistic controller. They tested their results by using a 3D simulator. Rusu et al. (2007) ubiquitous robotics is considered in the face of robots acting autonomously to build a mobile kitchen robot capable of setting table. Sensors and computing devices e.g. Radio frequency identification (RFID) tags are embedded into objects in a way that they connect automatically to each other to exchange information via integrating themselves to infrastructure as a part of it, even though they are not connected physically to each other. Before they implemented kitchen robot in real world, it was simulated in a virtual world. In another conspicuous research, Korayem et al. (2010) exert investigation on the maximum load capacity of a robot’s arm. They have implemented mobile two-links planar robot and a fixed 6R manipulator with complicated dynamic equations. Dynamic modeling of both cases is presented using the Lagrange method. Afterwards, the Dynamic Load Carrying Capacity (DLCC) is calculated for these two systems, thereafter, simulations are done for both cases and the DLCC of manipulators is determined. It is worthwhile to mention that the results were not simulated in a 3D environment. Ciocearl et al. (2010) have presented a software architecture for grasping of household objects. Using tactile sensors they have combined 3D scene interpretation, grasp planning, motion planning, and grasp failure identification and recovery. A clear-cut attribute pertinent to their work is the tight coupling between perception and manipulation, aiming to address uncertainties due to errors caused by sensor and execution. Their results illustrate that consistent performance can be achieved by breaking down a complex task into manageable bits. Skrzypczyk and Pieronczyk (2010) have studied autonomous vacuum cleaners; they have proposed a system designated for sweeping large and complex spaces using a team of cleaning robots. They have also discussed the difficulties pertinent to coordination of activities undertaken by individual robotic-unit; to resolve this issue they have presented application of non-cooperative Game Theory. Moreover, they have introduced hybrid architecture of control system and have applied the Nash equilibrium concept to generate a solution for the problem; the whole process was simulated in a 2D environment for various workspaces with different dimensions.

A plethora of techniques have hitherto been proposed to cope with the findspace problem (Brady, 1982; Latombe, 1991; Voros, 2001). Dealing with findpath problem, Ong and Gilbert (1998) suggest using penetration growth distance, a new measure for the depth of intersection between a pair of object models; while, Hornik et al. (1990) uses multilayered feed-forward networks. Strict financial requirements of the motion-planning, besides, exponential growth of analysis caused by larger number of nodes oblige using algorithms parallel in nature. Of these parallel algorithms, Neural Networks have been adopted by a number of authors to effectively reduce the tedious and lengthy computation (Kung and Hwang, 1989; Katic and Vukobratovic, 1994). Muñiz et al. (1995) through an unsupervised learning introduce an obstacle avoidance module integrated into the neural controller. This module makes use of sensory information to recursively determine a desired angle and distance that lead the robot to destination by avoiding the obstacles with various configurations, such as corridors, mazes, and doors. Obstacle avoidance is facilitated using Gaussian functions that avoid problems with local minima. Chohra et al. (1995) propose a neural navigation approach essentially based on pattern classification to acquire target localization and obstacle avoidance behaviors that treat the bot with intelligent navigation in a partially structured environments. Posterior to supervised Gradient Back-propagation learning, this approach provides vehicles with capability to recognize targets and obstacles. Decision-making and action engage two association stages, carried out by reinforcement Trial and Error learning which is followed by Neural Networks to identify the coordination. Fierroand Lewis (1998) introduced a control structure that integrates a kinematic controller with a Neural Network for non-holonomic mobile robots. This Neural Network controller can deal with un-modeled bounded disturbances and/or unstructured un-modeled dynamics in the vehicle. Fuji et al. (1998) provided the collision avoidance methods in a multi-robot structure using information exchanged through an infrared sensory system. In order to optimize the required computational efforts, they have developed a multilayered reinforcement learning scheme to attain accurate collision avoidance behaviors.

The rest of this paper is organized as follows: In Section II, a short introducing of Scout robot will be discussed then dynamic and kinematic models of mobile manipulator will be addressed. In the subsequent part of this section the optimal path for mobile manipulator is presented. In Section III, step-wise procedure undertaken to integrate artificial intelligence into the robot will be illustrated. In Section IV, simulation scheme of the robot has been elucidated. And finally we conclude this paper.
Mobile Manipulator (Scout)

In favor of practicality, an actual robot has been adopted in this study. Scout (Fig.1), a real-time robot, has been modeled in the course of simulation process with respect to the data-sheet provided by the manufacturer. Manufactured by Dr. Robot, Scout is an assembled wheeled WiFi robot that combines mobility and ability to grasp and manipulate objects, aimed to perform a variety of tasks. Scout is equipped with two gripping arms mounted on a chassis driven by two parallel non-holonomic wheels and a castor-wheel to preserve the balance for the bot. Each of the driver-wheels are equipped with 12V motor with over 300oz./inch of torque which makes the bot travel at a top speed of 75 cm/sec. There is a bumper mounted on the base which is equipped with shocks in case of a collision. On the wrist of right arm, there is a camera which facilitates remote manipulation of objects and improves robot’s ability to survey its surroundings. Each of the robot’s arms has 5 degrees of freedom consisting of three links to perform pitching, rolling and grasping. Either of the grippers have lifting capacity of 300 g, while, the chassis has a carrying capacity of 6 kg. Scout is controlled by a PC via a wireless connection (11Mbps; WiFi802.11) with dual serial communication channels. All the data and streaming between the robot and the PC pass at rates exceeding 10Hz via wireless link which allows to stream video up to 4 fps, sufficient to control the robot in real-time. There is also a speaker and a microphone onboard that enable a user to talk and hear via the robot. In addition, Scout has 6 infrared and 3 ultrasonic sensors. Respecting software, Scout includes all WiRobot development software components (MS Windows 2000 and up), enabling easy access to all data and information in a standard Microsoft Windows programming environment (e.g., MS VB and VC++).

Kinematic and Dynamic Equations

As mentioned previously the Scout robot platform has two driving and some castor wheels (Fig. 2). The problem of finding the Kinematic and dynamic model of non-holonomic Wheeled Mobile Manipulator has been treated by some authors before. According to Korayem et al. (2010) there were three constraints regarding mobile platform; one is about moving the platform in direction of axis of symmetry, and the other two are related to lack of slippage in the driving wheels.

\[
\dot{y}_c \cos \phi - \dot{x}_c \sin \phi - d \dot{\phi} = 0 \\
\dot{x}_c \cos \phi + \dot{y}_c \sin \phi + b \dot{\phi} = r \dot{\theta}_r
\]
Generalized coordinates of $q$ can be addressed as
$$
\begin{bmatrix}
\sin \phi & \cos \phi & -d & 0 & 0 & 0 \\
-cos \phi & -\sin \phi & -b & r & 0 & 0 \\
-cos \phi & -\sin \phi & b & 0 & r & 0
\end{bmatrix}
$$

The Lagrange equations of motion was used for platform, the general form of dynamic model for non-holonomic WMM can be introduced as
$$
\dot{V}(q,\dot{q}) = E(q)\tau - A^T(q)
$$
where $\lambda$ indicates the vector of the Lagrangian multipliers according to kinematic constraints and (Korayem et al., 2010):
$$
\dot{q} = Sv
$$
$$
S = \begin{bmatrix}
\frac{c(b \cos \phi - d \sin \phi)}{c(\cos \phi + d \sin \phi)} & \frac{c(b \sin \phi + d \cos \phi)}{c(\cos \phi - d \sin \phi)}
\end{bmatrix}
$$
At Eq.10, The first and second parts of the Lagrangian equation are multiplied by $S^T$.

Finally the kinematic and dynamic model of mobile manipulator can be introduced by Eq.11 (Korayem et al., 2010).
$$
\dot{x} = \left[(S^TMS)^{-1}(-S^TMS\dot{V} - S^T\dot{V})\right] + \left[(S^TMS)^{-1}\right]\tau
$$

**Optimal Path**

Optimal path is the term that resembles a state in which the robot requires minimum torque and kinetic energy to operate. According to Azimi (2010), optimal path for the mobile robot are represented in Eq.12. Control path and principle of minimum pontryagin are used for producing the optimal path equation and for decreasing the cost function.
$$
J = \frac{1}{2}(x^T \omega x + u^T R u)
$$

First Hamiltonian function was defined as it is mentioned below (Azimi, 2010):
$$
H = J + \psi \dot{x}
$$
Where $J$, the objective function is defined in the previous section and, $\dot{x}$ bears the state variables and $\psi$ is a vector which includes pseudo state variables. The number of $\psi$’s parameters are equal to number of state parameters. The optimal path can be found by using the functions declared beneath (Azimi, 2010):
$$
\dot{x} = \frac{\delta}{\delta u} + \frac{\delta}{\delta \dot{u}} = 0, \psi = -\frac{\delta H}{\delta x}
$$

Thereby, 14 differential nonlinear equations can be generated using the aforementioned functions. Owing to these correlations, robot’s optimal path can be determined respecting the cost function (Morgan, 2008).
MATERIALS AND METHODS

Construction of Training Data, Training Algorithm and Motion-planning Algorithm

The objective of the motion-planning problem in robotics can be declared as finding a continuous path from the initial position of the robot to the goal position considering robot’s initial location/orientation, location and orientation of the destination, and a set of obstacles located in a workspace. In other words, solving motion-planning problem empowers robot to avoid collisions with obstacles located on its route to the goal; this problem is divided into two sub-problems, called ‘Findspace’ and ‘Findpath’ problem. The findspace problem can be explained as construction of the configuration space containing the robot and the obstacles; whereby, the findpath problem can be described as preparation of a collision-free path for the robots from a given start position to the destination.

In this study, aside from considerations regarding the optimal control of the robot, its interaction with the surrounding environment is matched with that of humans. This situation is considered as if a blindfold human tries to move along a path in an unknown environment. In this condition, finding path is facilitated through sensing by means of hands. It is obvious that only those objects can be sensed which are within the vicinity of the human. As for a robot, hands can be resembled by ultrasonic sensors which have definite sensing capabilities. Accordingly, robot learns its environment – FIndspace problem – utilizing the training data which is acquired from the ultrasonic sensors. The attained data is then adopted by robot to autonomously devise an obstacle-free path to reach the desired destination – Findpath. To justify this unallied trait by robots, it requires feigning decision making structure. Hence, in favor of implementing artificial intelligence in the robot, two Neural Logic Networks or Neulonets – an artificial neural network – are exploited in this study. These neulonets are indexed as Primitive and Secondary neural networks.

In view of constructing the training data for the aforesaid Secondary neural network, a Primitive neural network is designed. Throughout the execution of this neural network, various trajectories with diverse initial and goal coordination gets constructed by dint of robot’s optimal control; withal, the acquired data from robot’s environment also account for the initial inputs of the neurons which encompass assortment of sensed obstacles along with conformance of robot’s and destination’s orientations. Availing from the inputs of the network, the output is determined revealing the direction toward which the robot will approach. In the midst of solving the motion-planning problem, iterative property of the neural network empowers the robot to update the synaptic weights until the outputs of the neural network converge to the desired results attained from the optimal control; hereby, the robot learns an internal model of the workspace by recurrent neural network. Thence, all the constructed inputs, outputs, and synaptic weights relevant to varied paths get stored to be recruited as training data in the Secondary neural network. Bearing both ‘obstacle-free environments’ and ‘workspaces with obstacles’, the Primitive algorithm for autonomous learning meant for alleviation of the motion-planning is structured including the following steps:

1. Retrieve ‘co-ordination and orientation of the initial points’, ‘co-ordination and orientation of the goal points’, and ‘co-ordination and dimensions of the obstacles’ placed in the environment.
2. Determine the optimal path intersecting any of the initial and goal points imposing optimal control equations.
3. Assess the presence or absence of obstacles within the vicinity of the bot via activation of the range finders (sensors).
4. In case of absence of obstacles, move robot along the path determined by optimal control until it either detects an obstacle or reaches the destination.
5. Based upon presumed coordinate-axes, calculate the slope of the line intersecting current and following positions along with the range of the sensors; then, determine the distances to the next position for any of the range finders using Pythagorean equations. Store the calculations in variable ‘l’.
6. At each new position, recruiting the findspce and findpath algorithms, determine and store the values for V, S, and O matrices – attributed to ‘findexpace’, ‘findpath’, and ‘direction of the next step’, respectively. In case of presence of an obstacle in l and S direction, execute 7th step, else move to the 4th step.
7. Recall V, S, and O matrices to be inserted as inputs for the neural network; update synaptic weights till the output of neural network converges to the desired results (Error rate of 0.1 % and 200 epochs has been considered for this algorithm).
8. Determine matrix O’s element Oα(a = 1,...,n), which is equal to 1. If the calculated output Oa differs in orientation with the preceding step, store ‘a’ – number of element in matrix O – in ‘row’ variable; else, retain the previous number in the ‘row’ variable.
9. In order to enhance the accuracy of the Primitive neural network, positioned at robot’s vicinity, hypothetical points are considered within each of the sectors. The coordination and orientation of this point is calculated according to the ensuing equations.

\[
\text{deg} = \frac{L}{2}
\]

(15)
Else $\rightarrow deg = f - 10 \times \left(\frac{\pi}{180}\right)$ (16)

$$x_1 = x + \left(\frac{d+1}{10}\right) \times \cos(\theta_1 + (row \times f) - deg)$$ (17)

$$y_1 = y + \left(\frac{d+1}{10}\right) \times \sin(\theta_1 + (row \times f) - deg)$$ (18)

$$\theta_2 = 2\pi - \left(\theta_1 + (row \times f) - deg\right)$$ (19)

where ‘the range of the sensors’, ‘angle of each sector’, and ‘degree of the hypothetical point’ are denoted by $d$, $f$, and $deg$ respectively.

Follow the route calculated in 8th step after computing the distance to destination and $\theta_2$ along with assessing presence or absence of obstacles within the current position and the hypothetical point.

Reiterate the 8th step to generate robot’s succeeding motion path until the bot reaches to an obstacle-free segment. In this case, move to the 4th step until either of the termination conditions is met.

After termination, store all the values for the matrices $V$, $S$, and $O$ to be used as training data for the Secondary neuralnetwork.

**Inputs and Outputs of the Neural Networks**

On the verge of satisfying the objective of autonomous motion-planning, a hybrid method, i.e. Neulonets were adopted to simultaneously conduct algorithms in solving the findspace and findpath problems, assisted with the sensed data acquired from the range finders. Toward achievement of this resolution, segments of workspace were devoted for each set of the sensors $i \ (i \in 1, \ldots, n)$, providing structure of matrix $V$’s elements, $V_i$’s. For the same number of sensors, matrix $S$’s elements, $S_i$’s were constructed.

All the objects in reach of robot’s sensors can be detected ranging from 0° to 360° in robot’s space, viz., an area enclosed by a circle of radius $d$ – range of sensors. This allows inputs imposition as normalized data acquired from ultrasound range finders. The findspace algorithm designed to identify elements of matrix $V$, $V_i$’s, recognizes the surrounding space formed of ‘n’ number of sectors (Fig. 6). For this purpose, multiple lines set angularly apart by 0.001 degrees are assumed to be positioned within the space of each of the sectors. Based upon presumed coordination-axes, the length of the aforesaid lines are calculated using the following equations:

$$x_1 = x + (d \times \cos(\beta))$$ (20)

$$y_1 = y + (d \times \sin(\beta))$$ (21)

where $d$ represent the range of the sensors and $\beta$ denotes each line’s angle.

In addition, in order to boost the accuracy of the range finders, half the lengths of the indicated lines are also calculated based upon the presumed coordination-axes.

$$x_2 = x + \left(\frac{d}{2} \times \cos(\beta)\right)$$ (22)

$$y_2 = y + \left(\frac{d}{2} \times \sin(\beta)\right)$$ (23)

The coordination of the apex and midpoint of each of the lines are then compared to the coordination of the obstacles. If any of the lines of a sector surpass the coordination of an obstacle, or if the calculated amounts fall behind the presumed minimum line-lengths, that sector gets devoted value 0 for its matrix $V$; in that, that segment is occupied and the motion within that segment is not granted. The contrary, $V_i = 1$ signalizes motion in this segment is possible due to absence of obstacles.

**Two-dimensional (2D) Modeling of the Robot**

The main resolution in developing a 2D simulator was to study and validate robot’s navigation, localization and mapping algorithms taking on the MATLAB environment prior to 3D simulation of the bot. MATLAB is the interactive programming software and is mostly used for doing computational tasks on high level mathematics. However, it has also been used in various fields like science, engineering, etc. MATLAB and its tool-boxes represent a very powerful tool for developing all kinds of algorithms. Aside from modeling of the kinematics, dynamics, and optimal-path of the manipulator, benefiting from MATLAB, models of the manipulator, wheels, range finders and the environment encompassing obstacles and the goal position was also introduced in the simulation. As sated previously, the purpose of the modeling was to create a simulation model to testify the results of motion-planning algorithm. Hence, a two-dimensional stationery environment was assumed in which robot autonomously travelled the planned route from start to goal position without having any collisions with the surrounding objects (Figure 9).
Posterior to construction of the outputs, the generated data from the Primitive neural network is then used as training data for the Secondary neural network. In the course of this study, a Secondary neural network is designed in favor of compromising the autonomous motion planning for the robot. Throughout this algorithm, recurrent neural network predicts succession of sensors inputs and on the base of the model it generates navigation steps as output commands. Iterative property of the neural network empowers the robot to store the

S: An index to identify whether the orientation is toward the goal or not.

V: An index to determine existence of an obstacle.

O: Robot’s orientation.

Figure 3. A schematic diagram of constructing training data for the mobile robot.
last orientation of the movement and to select the direction of the next navigation step in each of the iterations. Comprised of the ensuing steps, this algorithm facilitates stepwise and discrete series of motions for the robot to travel in a completely unknown workspace:

- Train Secondary neural network via acquisition of the training data prepared in the course of the Primitive neural network.
- Retrieve ‘coordination and orientation of the initial point’ along with ‘coordination and orientation of the destination’.
- Determine the optimal path intersecting the initial and destination positions via imposition of optimal control equations.
- Activate range finders to capture portions of the workspace at reach to determine presence or absence of obstacles.
- Follow the course of the optimal path determined by the optimal control as long as it either detects an obstacle or reaches the destination.
- Calculate the range of the sensors then determine the distances to the next position for any of the range finders using Pythagorean equations. Store the result in variable ‘I’.

At each new position, utilizing the findspace and findpath algorithms, determine and store the values for V, S, and O matrices. In case of presence of an obstacle in I and S direction, execute 8th step, else move to the 5th step.

- Recall V and S matrices to be inserted as inputs for the neural network; the output of this neural network signifies the next motion step.
- Determine matrix O’s element O_b (b ∈ 1,…,n), which is equal to 1. If the calculated output O_b differs in orientation with the preceding step, store ‘b’ – number of element in matrix O – in ‘row’ variable; else, retain the previous number in the ‘row’ variable.

Using the aforementioned equations (16, 17, and 18), the coordination and orientation of a hypothetical point positioned at robot’s vicinity is calculated in favor of enhancing the accuracy of the Secondary neural network.

In case of absence of obstacles within the current and hypothetical points, move the bot along the route calculated in the 9th step after setting values for \( \theta_2 \) (degrees to the goal point) and matrices S and V.

- Reiterate the 8th step to generate robot’s succeeding motion path until the bot reaches to an obstacle-free segment. In this case, move to the 5th step until either of the termination conditions is met.

**Three-dimensional (3D) Simulation of the Robot**

Succeeding the 2D simulation of the manipulator in MATLAB environment, a three-dimensional model of the robot was prepared using a variety of softwares. In pursuance of realization of this mark, conjointly with the already prepared model of the kinematics, dynamics, and optimal-path of the manipulator, a 3D simulation of the Scout robot was prepared exploiting the 3ds Max® software. In the course of 3D simulation, it was intended to use the least possible number of meshes in favor of imposing the least possible amount of computation overhead on the simulation engine. The constructed algorithm for the autonomous motion-planning of the bot was implemented to the 3D model through insertion of .dll files to the root directory of the 3D simulator. Hereby, a free space bearing the 3D model of the Scout robot was constructed in accord with its aforementioned actual physical specifications. Subsequently, the Scout robot autonomously advanced along the planned route from the initial position to the destination without having any collisions with the obstacles (Fig. 10). These results augment the soundness of the algorithm which was prepared in the course of this study for the motion-planning of the robot. Aside from validation of the results of this study, the 3D simulation might also empower practitioners to interpret robot’s behavior prior to observing it in action in the real-world.
On the verge of analyzing the motion-planning of the manipulator, neural network is recruited considering two of its most important capabilities, namely, their ability to learn and their good performance for the approximation of nonlinear functions. As mentioned previously, Neural Logic Network or Neulonet – an artificial neural network – is opted in this study which is a hybrid of neural network expert systems. A Neulonet has an ordered pair of numbers associated with each node and arrow; these ordered pairs take one of three values, namely, (1,0), (0,1) and (0,0). The topology of this network is illustrated in Fig. 5 which has a two layer structure; the input layer, a hidden layer and an output layer. There are ‘2n’ neurons placed on both the input layer and the first layer; besides, there are ‘n’ number of neurons planted on the second layer. The inputs of this network get constructed via findspace and findpath algorithms. The findspace algorithm is designed to assess presence or absence of obstacles as well as their coordination; the results of the findspace algorithm gets recorded in the matrix V. Withal, the findpath algorithm is developed to determine the shortest path oriented toward the destination and to store the results in the matrix S. The matrices V, S, and O are all
considered to have ‘n’ number of elements with regard to the number of sensors, ‘n’. Each of the neurons placed on the input layer are associated with the $S_i$’s and $V_i$’s; whereas, the neurons placed on the output layer are associated with the $O_i$’s which determines robot’s movement direction.

Figure 5. Topology of MLP network.

Besides, amid designing the findpath algorithm, the resolution is set to search for the shortest path oriented toward the destination. Accordingly, elements of the matrix $S$ get calculated via determination of the distances from each sensor $(x, y)$ to the goal position $(x_e, y_e)$, after setting the slope of the line passing through initial and goal points ($m$).

$$ m = \frac{(y_e-y)}{(x_e-x)} $$

$$ \forall \ i \in (1, ..., n) \begin{cases} x_i = x + (d \times \cos(\theta + (i \times f))) \\ y_i = y + (d \times \sin(\theta + (i \times f))) \end{cases} $$

Thereafter, for each of the sectors, summation of the distances from ending-points of radii to the goal point is assessed (Fig.7).

$$ d_i = \sqrt{(x_e-x_i)^2 + (y_e-y_i)^2} $$

$$ d_i^{i+1} = d_i + d_i^{i+1} $$

Thence, the segment with the least total amount receives value 1 for matrix $S$. After denoting the $S_i=1$ segment, all the other segments get treated with $S_i$’s equal to 0, which either proves deviations from the goal coordinates or reveals lengthy routes. To wit, matrix $S$ signifies whether the robot is situated toward the goal coordinates or not.

Figure 7. Determination of the distances from ending-points of radii to the goal point in each sector.

Succeeding setting elements of the input matrices $V$ and $S$, an output matrix, $O$, gets calculated to determine robot movement direction. Prior to assigning elements of this matrix, $O_i$’s, bisectors of each of the segments are assessed, meaning, designation of $i$ number of bisectors addressed for corresponding sensors.
In the course of each step, value 0 is devoted for any of the elements of this matrix except for the sensor $C(\epsilon_1, ..., \eta_1)$ holding $V_{c}=1$ and $S_{c}=1$ which receives value 1 for the element $O_{c}$. Consequently, exploiting the output of the neural network, the amount of angular distance between robot's current direction and the selected bisector gets calculated; consequently, robot swirls by the angular distance calculated and determines its azimuth by devising a route in direction of the bisector of the segment bearing $O_{i}=1$.

![Figure 8](image)

**Figure 8.** Robot's azimuth is calculated toward the bisector of the segment with $O_{i}=1$.

![Figure 9](image)

**Figure 9.** Robot travelling on the optimal-path without any contacts with the obstacles.

![Figure 10](image)

**Figure 10.** 3D simulation of collision-free movement of the Scout robot on the optimal-path.

**RESULTS AND DISCUSSION**

In this study, a simulation model of an actual robot hailed as Scout was treated with artificial intelligence to address the autonomous motion-planning problem by means of the Neural Networks. The optimal path for the base of the manipulator robot was formulated with the aid of the Kinematic and Dynamic functions alongside the Path profile; withal, Optimal Control was also utilized before applying Pontryagin's minimum principle which was coded in a numerical computing environment. Benefitting from a number of commercial softwares, behavioral investigation of the robot through various scenarios was facilitated. Thereafter, a 2D simulator was developed to study and validate robot's navigation, localization and mapping algorithms taking on the MATLAB environment prior to 3D simulation of the bot. Hence, in order to testify the results of motion-planning algorithm, a two-dimensional stationery environment was assumed within which the robot autonomously travelled the planned route from start to goal position with no collisions. Followed by 2D modeling, a 3D simulation was also taken...
place; results obtained from 3D simulation further guaranteed the soundness of the motion-planning algorithm of the robot. Aside from validation of the results of this study, the 3D simulation is envisaged to assist practitioners in interpreting robot’s behavior prior to any actual exertions in the real-world. At length, the findings of this research warrant that enhancing robot with the Kinematic and Dynamic functions, Path profile, Optimal Control and Artificial Neural Network would deliver autonomous behavior by the robot which promotes collision-free interactions.

Despite potencies of the proposed model, a need remain for methods that incorporate dynamic environment with moving obstacles instead of a static environment. Moreover, a comprehensive research extending evolutionary algorithms (EAs) to the proposed Neural Network module is an investigation area that deserves further devotion, which might further optimize the robot’s motion-planning.

REFERENCES


