Proceedings of the 2016 IEEE
International Conference on Robotics and Biomimetics
Qingdao, China, December 3-7, 2016

Path Planning with the Leapfrog Method in the Presence of Obstacles*

Belinda Matebese[1], Daniel Withey[2] and Mapundi K. Banda[3]

Abstract— **In this work, the Leapfrog algorithm from optimal control is presented as a method for optimal path planning for a mobile robot in the presence of obstacles. The proposed algorithm allows the robot to plan a collision-free path through static obstacles by minimizing a cost functional that includes energy terms and the Gaussian potential function. The Leapfrog path is initialized using the RRT planning algorithm and refines the RRT result to produce an optimal path.**

Comparison is made with the BVP4C optimization algorithm showing that similar path cost can be obtained with the Leapfrog approach. The Leapfrog algorithm shows value for continued development as an optimal path planning method since it initializes easily, creates a feasible path on each iteration, and can find solutions where other solvers may fail.

I. INTRODUCTION

Path planning for a mobile robot involves finding a route from a given starting state to a given goal state while avoiding obstacles in the environment. Common approaches include sampling-based methods, grid-based methods, and artificial potential fields [8].

Probabilistic roadmaps [1] and rapidly-exploring random trees (RRT's) [2]-[3] are examples of sampling-based methods, and the well-known A* algorithm [4] and its variations [5] are examples of grid-based methods. Artificial potential fields [6] - [7] can also be used to direct the robot by providing a function over the state space, the gradient of which defines state-dependent motion vectors to be applied to the robot to move it past obstacles and towards the goal.

In their simplest forms, however, these methods do not naturally include the robot kinematics and dynamics and therefore are mainly applicable to holonomic robots which can handle straight-line trajectories between arbitrary states. Nonholonomic motion planning [8] is an active area of research that explores methods applicable to a wider range of robots, such as those with car-like steering and those with differential-drive constraints.

In the framework of obstacle avoidance, the use of artificial potential fields was proposed by Khatib [6]. This approach has gained popularity in path planning for mobile robots due to its mathematical simplicity and elegance. Various works on artificial potential fields have been proposed in [9] - [10] to extend the initial concept. The goal state is supposedly the global minimum of the artificial potential field. However, the main drawback of the method is that it is sensitive to local minima, i.e., the robot can become trapped at a point far from the target.

To avoid this issue different functions like navigation potential-based functions, the Gaussian function have been used as potential-based functions. In [11] the navigation function was introduced. Unlike other potential field methods, the navigation function has only one minimum point (at the goal) so that there are no local minima. In Ren [12] a Gaussian function was used to model the attractor and a high order Gaussian-like function to model obstacles in order to avoid local minima. In [13] the Gauss function and a modified simulated annealing method was implemented for obstacle avoidance of multi-link robots. Korayem [14] presented an optimal motion planning approach for nonholonomic mobile robots in the presence of multiple obstacles based on the potential field method. The authors applied the potential function to the index performance of the optimal problem, to avoid the potential field limitations.

In path planning for mobile robots it is also important to generate an optimal path by optimizing a performance criterion such as distance, time or energy. Hence an optimal control problem needs to be solved i.e. one needs to find a control law for a given system such that a requisite optimality criterion is achieved. A solution to a set of differential equations describing the paths determined by the control variables that also minimizes a cost functional which is a function of a state and control variables is determined. Optimal control problems are generally nonlinear hence in most cases analytical solutions are untenable. Numerical methods are, therefore, used to approximate their solutions.

The numerical methods are classified as direct or indirect methods. Direct methods are based on nonlinear programming (NLP) [15] whereas indirect methods are based on application of Pontryagin's minimum principle (PMP) [16]. The commonly used indirect methods are the collocation method, shooting and multiple-shooting methods. Recently, in optimal path planning of different types of mobile robots indirect methods have been of interest, see [17] - [19], in which the *bvp4c* package in MATLAB [20], which is based on the collocation method was used to solve differential systems to find optimal trajectories.

An indirect-based method called the Leapfrog algorithm was introduced in [21]. Similar to multiple shooting, in the Leapfrog method the time interval is subdivided and local optimal controls are found separately over each subinterval. The subintervals of optimal control trajectories are updated through a scheme of midpoint maps. A significant difference

*This research is supported by the Council for Scientific and Industrial Research (CSIR). It is also supported in part by the National Research Foundation of South Africa (Grant number: 93099)

[1,2] B.T. Matebese and D.J. Withey are with the Mobile Intelligent Autonomous Systems (MIAS) unit at the Council for Scientific and Industrial Research (CSIR), Pretoria, South Africa {bmatebese, dwithey}@csir.co.za

[3] M.K. (Mapundi) Banda is with the Department of Mathematics and Applied Mathematics, University of Pretoria, Pretoria, South Africa mapundi.banda@up.ac.za

is that leapfrog uses overlapping subintervals. In addition, the method does not depend on the provision of good initial guesses along a path and the optimal solution provided by the method satisfies both boundary conditions at every step. For each iteration, state trajectories are feasible, which is not generally the case with multiple shooting approaches. The feasible trajectories can be improved continually obtaining sub-optimal trajectories. This might be advantageous when time is critically limited. Lastly the number of subdivisions along the feasible path can be adjusted hence the number of subdivisions is reduced as the solution is approached. This was demonstrated in [22] where numerical implementation of the Leapfrog algorithm for nonlinear systems was given. In our previous work [23], application of the Leapfrog method to mobile robot path planning was introduced. This was done by solving the mobile robot kinematic model as an optimal control problem to find optimal trajectories. The method showed to be efficient and capable of overcoming challenges faced by other numerical methods.

The present work represents the first application of the Leapfrog method to find optimal trajectories for mobile robot path planning in the presence of obstacles. Of particular interest in this present work is the use of the potential function added in the cost functional to avoid the potential function limitations as described in [14]. Hence the Gaussian potential function (GPF) was added to the cost functional as a collision avoidance parameter. Using PMP, necessary conditions for optimality are obtained and the Leapfrog method is used to numerically solve the resulting two-point boundary value problem (TPBVP). One of the requirements for the Leapfrog method is the feasible, initial partition. For this reason the Rapidly-exploring Random Trees (RRT) algorithm [8] has been employed to determine the initial feasible path.

The remainder of this paper is organized as follows: Section II describes the basic methods, Section III describes the cost function used in the optimal control approach, and Section IV describes the experiments and their results. Subsequently, Sections V and VI provide discussion and conclusions, respectively.

II. Methods

A. Mobile Robot Kinematics

The mobile robot considered here has two driving wheels mounted on the same axis with independent direct current (DC) motors and one free caster wheel. The model of the robot operating on a planar surface can be seen in Figure 1. The robot motion in a generalized coordinate system can be represented as $\mathbf{q} = (x,\, y,\, \theta)$, where (x_r, y_r) is the position, and θ is the orientation of the robot. Assuming that the mobile robot is moving on a plane without slipping (or tilting), the nonholonomic constraint represented by Equation (1) is also considered.

$$-\dot{x}\sin\theta + \dot{y}\cos\theta = 0. \tag{1}$$

From the nonholonomic constraints, the two-wheeled mobile robot kinematic model can be derived as

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}\cos\theta & 0\\ \sin\theta & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}v\\ \omega\end{bmatrix}, \tag{2}$$

where v and ω are the linear and angular velocities, respectively.

B. Gaussian Potential Function

Potential field methods provide a simple and elegant solution to many motion planning problems for mobile robots. In this work the Gaussian function is used to form the attractive and repulsive fields. This approach has advantages compared to other potential functions. One of the advantages is that the size of the attractive and repulsive force can be adjusted by adjusting the variance σ^2, and the parameter C which determines the effectiveness of the obstacle [12].

Figure 1 shows a representation of potential functions and two-wheeled mobile robot.

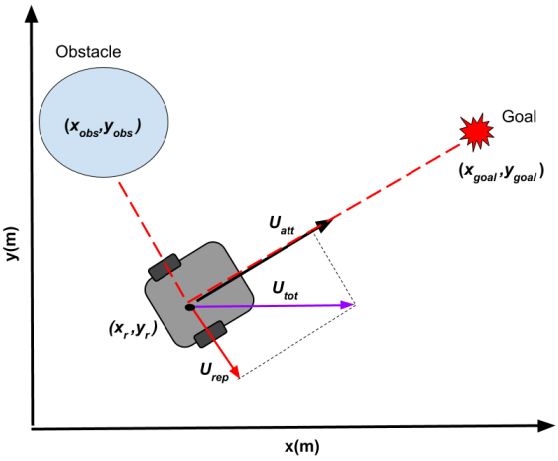

Fig. 1. Representation of potential forces in a 2D workspace.

The attractive force, U_{att}, that is produced by the goal can be expressed using the Gaussian function as

$$U_{\text{att}}(q) = 1 - e^{-\frac{1}{2}\left(\frac{d^2_{\text{goal}}}{\sigma^2_{\text{att}}}\right)}, \tag{3}$$

where σ_{att} is a constant that defines the width of the distribution. The parameter d_{goal} is defined as distance from the robot to the goal.

$$d_{\text{goal}}(q) = \sqrt{(x_r - x_{\text{goal}})^2 + (y_r - y_{\text{goal}})^2} \tag{4}$$

where $(x_{\text{goal}}, y_{\text{goal}})$ is the position of the goal. The repulsive force for obstacle j, U_{rep_j}, can be described as

$$U_{\text{rep}_j}(q) = A_{\text{rep}} e^{-\frac{1}{2}\left(\frac{d^2_{\text{obs}_j}}{\sigma^2_{\text{rep}_j}}\right)^C} \quad \text{for } j = 1, \ldots, n \tag{5}$$

where n is the number of obstacles, A_{rep} is a positive constant and d_{obs_j} is the distance between the robot and the $j-$th obstacle,

$$d_{\text{obs}_j}(q) = \sqrt{(x_r - x_{\text{obs}_j})^2 + (y_r - y_{\text{obs}_j})^2} \tag{6}$$

where $(x_{\text{obs}_j}, y_{\text{obs}_j})$ is the position of the j-th obstacle.

In Equations (4) and (6), (x_r, y_r) represent the robot's position.

Then the total potential force, U_{tot}, on the robot is therefore

$$U_{\text{tot}}(q) = U_{\text{att}}(q) + \sum_{j=1}^{n} U_{\text{rep}_j}(q); \tag{7}$$

Figure 2 shows the obstacle and a target represented by Gaussian potentials.

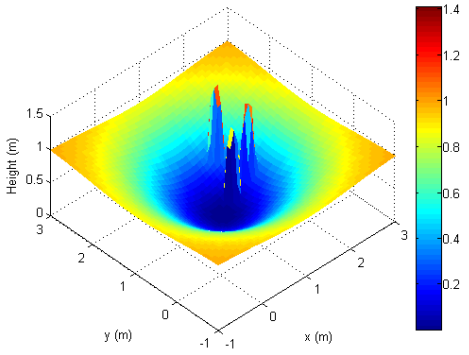

Fig. 2. Three obstacles and the target (dark blue) generated by the Gaussian function

C. Rapidly-exploring Random Tree Algorithm

The RRT algorithm is designed in such a way that a tree that expands randomly over the configuration space q is maintained. Firstly, the tree T is initialized with an initial node, q_{init}, the reader may refer to Figure 3. While the goal configuration is not reached, an iteration is performed in which at each step a random node q_{rand} is selected. To extend the tree, the nearest node q_{near} to q_{rand} that is already in the tree is selected. Moving a certain distance ϵ from q_{near} in the direction of q, check for any collision and create a new node, q_{new}.

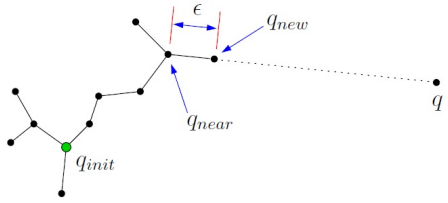

Fig. 3. RRT algorithm [24]

This process of selecting a node and extending the tree is repeated until a goal configuration is reached and added to the tree.

D. Optimal Control

Optimal control deals with a problem of finding a control law for a given system such that a certain optimality criterion is achieved.

A general optimal control system can be modelled by

$$\min_{u \in U} \int_{t_0}^{t_f} L(\mathbf{x}(t), \mathbf{u}(t))\, dt \tag{8}$$

subject to

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \tag{9}$$

$$\mathbf{x}(t_0) = \mathbf{x}_0,\ \mathbf{x}(t_f) = \mathbf{x}_f, \tag{10}$$

where $\mathbf{x}$ is the state vector, $\mathbf{u}$ is a control input, t_0 and t_f represent initial and final time (fixed) and $\mathbf{x}_0$, $\mathbf{x}_f$ are the initial and final state, respectively. Below, for brevity, t will be omitted from the equations. To solve an optimal control problem necessary conditions of optimality need to be formulated. Firstly, the Hamiltonian function is defined by adjoining the right hand side of (9) and cost function using the Lagrange multiplier, λ, as

$$H(\lambda, \mathbf{x}, \mathbf{u}) = L(\mathbf{x}, \mathbf{u}) + \lambda^T \mathbf{f}(\mathbf{x}, \mathbf{u}). \tag{11}$$

Using the Pontryagin Minimum Principle (PMP), necessary conditions for optimality are obtained as follows

$$\dot{\mathbf{x}} = \frac{\partial H}{\partial \lambda} \tag{12}$$

$$\dot{\lambda} = -\frac{\partial H}{\partial \mathbf{x}} \tag{13}$$

$$0 = \frac{\partial H}{\partial \mathbf{u}} \tag{14}$$

where (12) is equivalent to the state equation (9) and (13) is a costate equation. The control, $\mathbf{u}(t)$, which is a function of $\lambda(t)$ and $\mathbf{x}(t)$, is obtained from (14). The boundary conditions are defined as in (10).

E. Leapfrog Method

In this section, the Leapfrog algorithm is described for mobile robot path planning in the presence of obstacles. The RRT method is employed to generate an initial collision-free feasible path, P_z. In this work the initial and final states $z_0 = P_z(t_0)$, $z_q = P_z(t_f)$ and initial and final time t_0 and t_f are fixed. The feasible path P_z given from initial position to final position is then divided into q path segments such that $z_i = P_z(t_i)$, for $i = 1, \ldots, q-1$. In the k-th iteration, given z_{i-1}^k, z_{i+1}^k and t_{i-1}^k, a Subproblem (15) – (16) is solved.

$$\min_{u \in U} \int_{t_{i-1}}^{t_{i+1}} L(\mathbf{x}(t), \mathbf{u}(t))\, dt \tag{15}$$

subject to

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \tag{16}$$

$$\mathbf{x}(t_{i-1}) = \mathbf{z}_{i-1},\ \mathbf{x}(t_{i+1}) = \mathbf{z}_{i+1},$$

A midpoint $z_i^{k+1} = x(t_i)$ and $t_i^{k+1} = t_i$ is thereafter computed in such a way that the cost along an optimal path between z_{i-1}^k and z_{i+1}^k is halved. This is repeated until the maximum number of iterations is reached. In Fig. 4, the Leapfrog algorithm is illustrated for $(i = 1, 2, 3)$, and for

two iterations [21]. The solution of the Subproblem (15) – (16) and the associated optimality system is used to obtain updates of z_i together with updates for the costate λ_i and t_i.

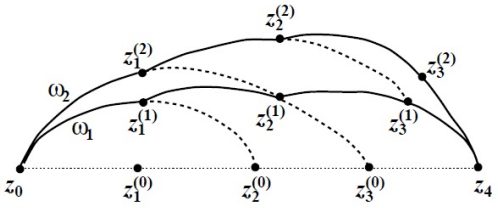

Fig. 4. Leapfrog Algorithm, illustration from [21]

III. Problem Formulation

In this section, an optimal control problem for a two-wheeled mobile robot navigating a region with obstacles is presented. The robot is assumed to plan a path in a 2D workspace. The aim is to plan a collision-free path for the mobile robot from an initial pose (x_0, y_0, θ_0) to a final pose (x_f, y_f, θ_f) (fixed). This will be achieved by finding a control $\mathbf{u}(t)$ which determines the mobile robot's collision-free trajectory.

In this work, the Gaussian potential function defined in (7) is used to avoid collisions between the robot and the obstacles. The cost functional is defined to minimize the energy control inputs which include the Gaussian potential function. The cost functional of the optimal control problem is, therefore, defined as

$$J = \frac{1}{2}\int_{t_0}^{t_f} [u(t)^T Q u(t) + U_{\text{rep}}(q)]\, dt. \tag{17}$$

where $u = [v, \omega]^T$, $U_{\text{rep}} = \sum_{j=1}^{n} U_{\text{rep}_j}$ is the total repulsive and Q is the control weighting matrix. Using the necessary conditions of optimality (12) and (13), and substituting the computed control from (14) in the state and costate equation yields the TPBVP presented in (18).

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\lambda_1} \\ \dot{\lambda_2} \\ \dot{\lambda_3} \end{bmatrix} = \begin{bmatrix} -\frac{1}{2}\lambda_1(1+\cos 2\theta) - \frac{1}{2}\lambda_2 \sin 2\theta \\ -\frac{1}{2}\lambda_1 \sin 2\theta - \frac{1}{2}\lambda_2(1-\cos 2\theta) \\ -\lambda_3 \\ \sum_{i=1}^{n} \nabla_x U_{\text{rep}_j} \\ \sum_{i=1}^{n} \nabla_y U_{\text{rep}_j} \\ \frac{1}{2}(\lambda_2^2 - \lambda_1^2)\sin 2\theta + \frac{1}{2}\lambda_1\lambda_2 \cos 2\theta \end{bmatrix} \tag{18}$$

where

$$\nabla_x U_{\text{rep}_j} = -\frac{1}{2\sigma} A_{\text{rep}} C(x - x_{\text{obs}}) \left(\frac{d_{\text{obs}}^2}{\sigma^2}\right)^{C-1} e^{-\frac{1}{2}(\frac{d_{\text{obs}}^2}{\sigma^2})^C}$$

$$\nabla_y U_{\text{rep}_j} = -\frac{1}{2\sigma} A_{\text{rep}} C(y - y_{\text{obs}}) \left(\frac{d_{\text{obs}}^2}{\sigma^2}\right)^{C-1} e^{-\frac{1}{2}(\frac{d_{\text{obs}}^2}{\sigma^2})^C}$$

To solve the above TPBVP in Equation (18), the boundary conditions (10) must be satisfied.

IV. Numerical Results

In this section, the optimal control problem for the mobile robot to navigate a space with obstacles is numerically solved using the Leapfrog method. Real experiments are conducted using the Pioneer 3DX mobile robot. For the simulations, an initial position plus orientation and a goal position plus orientation is prescribed. The mobile robot will plan its path in a workspace with circular obstacles located at center point $(x_{\text{obs}_j}, y_{obs_j})$ and radius r_{obs_j} for $j = 1, \ldots, n$. The position of the obstacles in the workspace is known to the robot. The control weighing matrix in cost function (17) is given by $Q = \text{diag}(1, 1)$. A path generated by the RRT algorithm was used as a feasible initial partition for the Leapfrog method.

The simulations were done using MATLAB. A MATLAB solver, BVP4C, which can solve optimal control problems efficiently [20] was also employed. In this work, the BVP4C solution is used for comparison to the Leapfrog solution. Both methods used same parameters described in TABLE I and same final time (t_f) per problem.

TABLE I
Gaussian Potential Parameters

height of repellent	$A_{\text{rep}} = 1$
width of repellent	σ_{rep}=r_{obs_j}
steepness of the obstacle	$C = 1$

A. Simulation Results

Case 1: In this simulation, the robot's prescribed initial state is $[0, 0, 0]$ and the final state is $[1.5, -0.4, 0]$ with a total time of $t_f = 2s$. The obstacle has a center point with coordinates $x_{\text{obs}_1} = 0.8$, $y_{\text{obs}_1} = 0$ and radius $r_{\text{obs}_1} = 0.1$. Figure 5 shows the trajectories generated by Leapfrog and BVP4C.

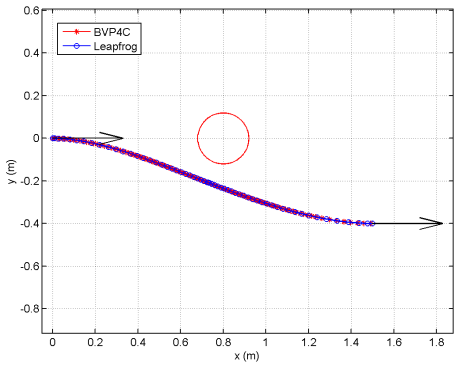

Fig. 5. Case 1: The optimal trajectories obtained from BVP4C and Leapfrog in the presence of one obstacle.

Case 2: For this simulation the mobile robot is desired to move from a given initial state $[0, 0, \pi/4]$ to a final

state $[1, 1, \pi/4]$ within total time $t_f = 4s$. The obstacles are located at center points with coordinates $x_{\text{obs}_1} = 0.35$, $y_{\text{obs}_1} = 0.45$ and $x_{\text{obs}_2} = 0.55$, $y_{\text{obs}_2} = 0.7$, respectively. The radii of the obstacles are $r_{\text{obs}_1} = 0.1$ and $r_{\text{obs}_2} = 0.1$, respectively. As it can be seen in Fig. 6, the Leapfrog method is able to plan an optimal collision free path.

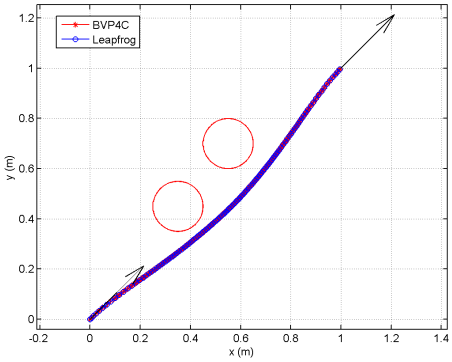

Fig. 6. Case 2: The BVP4C and Leapfrog optimal trajectories in the presence of two obstacles.

Case 3: In this simulation, the robot is moving from the initial state $[0, 0, 4\pi/9]$ to the final state $[1, 1, 4\pi/9]$ within a total time of $t_f = 5s$. The centre coordinates of the obstacles are $x_{\text{obs}_1} = 0.45$, $y_{\text{obs}_1} = 0.2$, $x_{\text{obs}_2} = 0.55$, $y_{\text{obs}_2} = 0.8$ and $x_{\text{obs}_3} = 0.8$, $y_{\text{obs}_3} = 0.25$ with the radii $r_{\text{obs}_1} = 0.06$, $r_{\text{obs}_2} = 0.06$ and $r_{\text{obs}_3} = 0.06$. In Fig. 7, it can be noted that the state trajectory generated by the BVP4C solver is very similar to the Leapfrog trajectory.

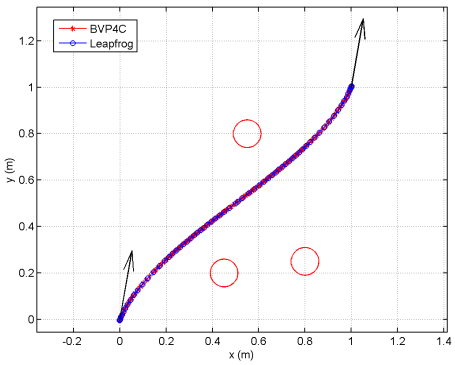

Fig. 7. Case 3: The optimal trajectories obtained from BVP4C and Leapfrog in the presence of three obstacles.

Case 4: Lastly, the robot is considered to move from the initial state $[0, 0, 0]$ to the final state $[2, 0, 0]$ within a time up to $t_f = 5s$. The center coordinates of the obstacles are $x_{\text{obs}_1} = 0.9$, $y_{\text{obs}_1} = 0$, $x_{\text{obs}_2} = 1.1$, $y_{\text{obs}_2} = 0$, $x_{\text{obs}_3} = 0.3$, $y_{\text{obs}_3} = -0.35$, $x_{\text{obs}_4} = 1.6$, $y_{\text{obs}_4} = 0.35$ and $x_{\text{obs}_5} = 0.4$, $y_{\text{obs}_5} = 0.3$ with the radii $r_{\text{obs}_i} = 0.08$, for $i = 1, \ldots, 5$. In Fig. 8, it can be seen that the Leapfrog algorithm managed to find a collision-free path whereas the BVP4C path cut through the obstacles.

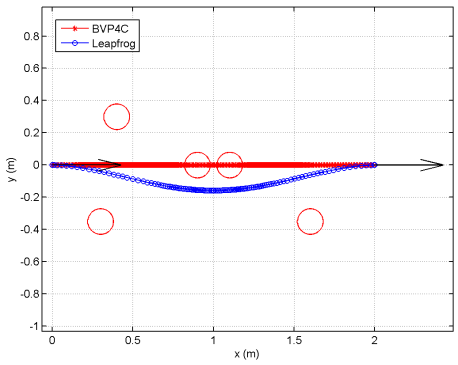

Fig. 8. Case 4: The optimal trajectories obtained from BVP4C and Leapfrog in the presence of five obstacles.

The path cost for both the Leapfrog and BVP4C were calculated to evaluate the efficiency of the methods. To compute the path cost, the forward and angular velocities together with the Gaussian function for each time point were used in the calculation of the integrand in (17). The resulting discrete-time function was then integrated numerically using *trapz*() in MATLAB. Table II shows the values of the path cost for the cases above.

TABLE II
PATH COST VALUES

Case	Leapfrog	BVP4C
1	0.8052	0.8052
2	0.5022	0.5022
3	0.5022	0.5022
4	0.6043	0.7023

B. Experiments Results

Path following experiments were conducted using a Pioneer 3DX mobile robot. A simple proportional plus derivative (PD) controller was implemented where the path generated by Leapfrog were taken as inputs. The PD controller then sends approximate velocities to the robot so that it follows the Leapfrog-computed position (x, y), and the sign of the Leapfrog-computed forward velocity.

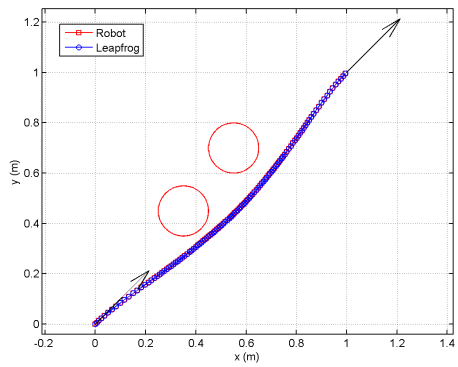

Fig. 9. Case 2: The optimal trajectories obtained from Pioneer robot and Leapfrog simulation in the presence of obstacles.

Table III shows the Mean Square Error (MSE) for the robot position relative to the Leapfrog position for all the cases. In Table III it can be noted that the results obtained

TABLE III
MSE VALUES FOR REAL EXPERIMENTS

Case	MSE (m^2)
1	2.1076e-06
2	1.03979e-06
3	3.2580e-06
4	1.9094e-06

from the robot experiments are close to those obtained from the Leapfrog simulations.

V. DISCUSSION

In Table II, it can be noted that the cost values for the Leapfrog method were very close to BVP4C cost values. This shows that the Leapfrog method has the capability to perform well even in the presence of obstacles. In [23] it was shown that the Leapfrog method can produce solutions in cases where BVP4C encountered singularities. Case 4 and Fig. 7 demonstrate that the Leapfrog managed to find a collision-free path where as for BVP4C the path went through the obstacles. From the case studies in Section IV above, it can be seen that the Leapfrog algorithm performs well in complex environments.

For the simulation experiments, a relatively low number of RRT waypoints were used for convergence efficiency. As the waypoints are selected randomly, the path generated each time the RRT algorithm executes is not the same. This means that the computational time to find a solution in Leapfrog may vary. Also, the Leapfrog algorithm is sensitive to the selection of the number of iterations and number of points per iteration, although sensitivity to parameter selection is also an issue for the BVP4C algorithm.

VI. CONCLUSIONS AND FUTURE WORK

This paper shows that the Leapfrog algorithm can generate optimal paths in the presence of obstacles and has the potential to be useful in optimal path planning for mobile robots. This was shown through simulation and experimental solutions. The Leapfrog method has advantages in that it initializes easily using a feasible, suboptimal path. In addition, it produces a drivable path for the robot on each iteration of its computation.

For future work, formulation of the initial partition using different path planners, for example, grid-based planners, will be investigated. Developments for the Leapfrog to be compared with conventional path planning methods will also be done. Another research direction is to extend the Leapfrog method to perform path planning for mobile manipulators. Application of the method in more complex obstacle environment will also be considered.

REFERENCES

[1] L. Kavraki, P. Svestka, J.-C. Latombe, M. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans. Robot. Autom. vol.12, no.4, pp. 566 - 580, 1996.

[2] S. Karaman, E. Frazzoli, Sampling-based algorithms for optimal motion planning. Int. J. Robot. Research, vol.30, no.7, pp. 846 - 894, 2011.

[3] S. LaValle, J. Kuffner, Randomized kinodynamic planning. Int. J. Robot. Research, vol. 20, no. 5, pp. 378 - 400, 2001.

[4] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall Englewood Cliffs. NJ, 2009.

[5] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, S. Thrun. Anytime search in dynamic graphs, Artificial Intelligence, vol. 172, no. 14, pp. 1613 -1643, 2008.

[6] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots. Int. J. Robot. Research, vol. 5, no. 1, pp. 90 - 98, 1986.

[7] E. Rimon and D.E. Koditschek, Exact robot navigation using artificial potential functions. IEEE Trans. Robot. Autom., vol. 8, no. 5, pp. 501 - 518, 1992.

[8] S.M. LaValle, Planning algorithms. Cambridge University Press, 2006.

[9] S.S. Ge and Y.J. Cui, New potential functions for mobile robot path planning. IEEE Trans. Robot. Autom., vol. 16, no. 5, pp. 615 - 620, 2000.

[10] J. Guo, Y. Gao and G. Cui, Path planning of Mobile Robot Based on Improved Potential Fields. Inf. Tech. J., vol. 12, no. 11, pp. 2188 - 2194, 2013.

[11] E. Rimon and D.E. Koditschek, Exact robot navigation using artificial potential functions. IEEE Trans. Robotic. Autom., 1992, vol. 8, no. 5, pp. 501 - 518.

[12] J. Ren and K.A. McIsaac, A Hybrid-Systems Approach to Potential Field Navigation for a Multi-Robot Team. IEEE Int. Conf. Robot. and Autom., 2003.

[13] D. Yagnik, J. Ren and R. Liscano, Motion Planning of a Multi-Link Robot With Artificial Potential Field Method and Modified Simulated Annealing. Proc. of 6th MESA Conf. Elect. Comput. Eng., pp. 421 - 427, 2010.

[14] M.H. Korayem, M. Nazemizadeh, H. Binabaji, and V. Azimirad, Optimal Motion Planning of Non-holonomic Mobile Robots in Presence of Multiple Obstacles. Int. Conf. Emerg. Trends in Robot. Comm. Tech., pp. 269 - 272, 2010.

[15] J.T. Betts, Practical Methods for Optimal Control Using Nonlinear Programming. SIAM J. Numer. Anal., 2001.

[16] L.S. Pontryagin, V.G. Boltianski, R.V. Gamkrelidze, E.F. Mishchenko, and D.E. Brown, The mathematical theory of optimal processes. London, Paris,a Pergamon Press book, 1964.

[17] M.H. Korayem, H.N. Rahimi, A. Nikoobin, Path planning of mobile elastic robotic arms by indirect approach of optimal control. Int. J. Adv. Robot. Syst., vol. 8, no.1, pp. 10 - 20, 2011.

[18] M.H. Korayem, M. Nazemizadeh and H.R. Nohooji, Optimal point to point motion planning of nonholonomic mobile robots in the presence of multiple obstacles. J Braz. Soc. Mech. Sci. Eng., vol. 36, pp. 221 - 232, 2014.

[19] A. Nikoobin, M.R. Vezvari and M Ahmadieh, Optimal Balancing of Planar Cable Robot in Point to Point Motion using the Indirect Approach. RSI Int. Conf. Robot. Mech., 2015.

[20] Mathworks, Matlab software description. [Online]. Available:http://www.mathworks.com, 2014.

[21] C. Kaya and J. Noakes, Geodesics and an optimal control algorithm. Proc. 36th IEEE Conf. Decis. Control (CDC), pp. 4918 - 4919, 1997.

[22] C. Kaya and J. Noakes, The leap-frog algorithm for optimal control. SIAM J. Numer. Anal., vol. 46, no. 6, pp. 2795 - 2817, 2008.

[23] B.T. Matebese, M.K. Banda and D.J. Withey, Application of the Leapfrog Method to Robot Path Planning. Proc. IEEE Int. Conf. Inf. Autom., 2014.

[24] J.Kuffner, S. LaValle, RRT-connect: An efficient approach to single-query path planning. IEEE Int. Conf. Robot. Autom., vol. 2, pp. 995 - 1001, 2000.

[25] D. Kirk, Optimal Control Theory. An Introduction. New York: Prentice Hall, Inc., 1998.