Proceedings of the 13th
Mediterranean Conference on Control and Automation
Limassol, Cyprus, June 27-29, 2005

Trajectory Tracking in the Presence of Obstacles using the Limit Cycle Navigation Method

Raphael Grech and Simon G. Fabri†
Department of Electrical Power and Control Engineering
University of Malta
Msida MSD 06, Malta
† *E-mail: sgfabr@eng.um.edu.mt*

***Abstract*— This paper proposes a system for effecting trajectory tracking in combination with obstacle avoidance in mobile robotic systems. In robotics research, these two situations are typically considered as separate problems. This work approaches the problem by integrating classical trajectory-following control schemes with Kim et al.'s Limit Cycle Navigation method for obstacle avoidance. The use of Artificial Potential Function methods for obstacle avoidance is purposely avoided so as to prevent the well-known problems of local minima associated with such schemes. The paper also addresses the problem of non-global obstacle sensing and proposes modifications to Kim et al.'s method for handling multiple, overlapping obstacles under local sensing conditions.**

I. Introduction

An important problem in mobile robotics deals with the design of a suitable control system for making a robot follow some desired trajectory in time, whilst simultaneously avoiding obstacles that might be located on the path defined by this trajectory. Classically, these two requirements have been solved as separate problems: obstacle-free tracking of a trajectory in one instance [1], or obstacle avoidance without trajectory constraints in the second instance [2], [3].

Control algorithms for handling situations that integrate these two sub-problems have been proposed by Grech and Fabri in [4], where the trajectory tracking algorithm proposed by Oriolo et al. [1] was combined with a modified version of the classical Artificial Potential Function (APF) method [2] for obstacle avoidance. In the presence of obstacles blocking the desired trajectory, the three methods proposed in [4] generate a new local path around the obstacle. The robot will therefore temporarily depart from the target trajectory so as to avoid an obstacle. Once the obstacle is circumvented, the robot will follow the desired trajectory until another obstacle is detected and the avoidance mechanism is repeated.

The APF method for obstacle avoidance suffers from the well-known problem of local minima. Given certain obstacle configurations, the robot could get stuck in a so-called potential well and the robot stops moving, thereby failing to circumvent the obstacle. An example of this problem is shown in Figure 1 where the robot is faced with a particularly challenging U-shaped obstacle configuration. The robot fails to circumvent the obstacle and is unable to catch up with the desired circular trajectory from that instant onwards.

In this paper, instead of relying on APF techniques for obstacle avoidance, we make use of the Limit Cycle Navigation method originally proposed by Kim et al. in [5]. This method overcomes the local minima problem associated with APF techniques by generating a new local trajectory based upon the limit cycle dynamics of a second order system. The circular trajectory of the proposed limit cycle helps in providing a smooth trajectory for the robot to follow. The Limit Cycle Navigation method has been adopted in this paper because it is a simple, yet effective way of producing an alternative trajectory around an obstacle. Also, it is not computationally intensive for real time processing.

Whereas the work of Kim et al. [5] addressed only the problem of obstacle avoidance, our main contribution in this paper is to combine the Limit Cycle Navigation method with trajectory tracking so that the robot continues to follow the desired trajectory after an obstacle is bypassed. This way we are able to avoid situations of the type shown in Figure 1.

Another important contribution of this paper is motivated by its application. This research is ultimately aimed towards the development of a mobile robotic system based in a factory shop floor to transport material to and from automated assembly machines by following some predefined desired

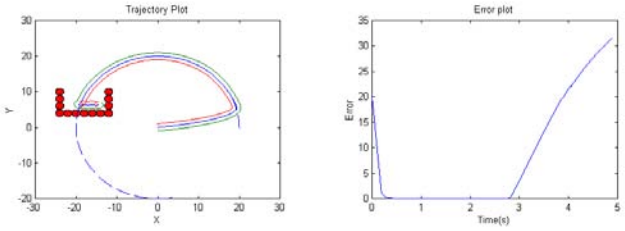

Fig. 1. APF-based obstacle avoidance: (a) robot trajectory stuck in a potential well (b) Euclidean norm of positional error

trajectory. In this kind of scenario, a global view of the whole shop floor to detect obstacles would be a costly implementation. Hence, in our system we are assuming that the robot can only sense obstacles located within a limited range in its vicinity. In addition, the environment in which the robot is operating is highly unstructured. This means that maps indicating obstacle locations are not really relevant because most obstacles are changing position continuously. Hence the robot has to circumvent obstacles in a reactive manner, *i.e.* it must have the capacity of avoiding an obstacle or a set of obstacles based only upon *local* information sensed in the robot's vicinity.

Due to the issues mentioned above, the integrated trajectory tracking and obstacle avoidance system being proposed in this paper is designed on the principles of local navigation. The restriction of having only local data about the environment poses an additional challenge for the robot to be able to circumvent obstacles without getting stuck in loops or local minima. Hence, the Limit Cycle Navigation method proposed in [5] is modified to cater not only for trajectory tracking, but also for the localized sensing of obstacles. These two features, which were not addressed in [5], are essential considerations for our application. In this paper we will show how the robot can navigate in cluttered environments and how it is capable of handling obstacle configurations where an APF approach would fail due to potential wells.

The paper is organised as follows: Section II gives some background on Trajectory Tracking Control and Kim et al.'s Limit Cycle Navigation method. Section III describes how we have modified the Limit Cycle Navigation method for the case of local sensing and Section IV describes how the Limit Cycle Navigation method for obstacle avoidance has been combined with Trajectory Tracking control. Section V shows some simulation results and finally, in Section VI, conclusions are drawn and future developments discussed.

II. Background

The robot considered in this report is a differentially driven, wheeled mobile robot (WMR). This type of robot is driven using two independently controlled wheels. The kinematic structure of the vehicle prohibits certain vehicle motions and imposes nonholonomic constraints [6]. The control inputs are the robot linear velocity ( ) and the angular velocity ( ). The kinematic state of the robot is given by vector $[\quad\quad]^T$, where (,) represent the Cartesian position of the robot on a plane with respect to some reference frame and is the robot orientation. The triplet (, ,) is often referred to as the pose of the robot. Using the kinematic model for a differentially driven WMR, the pose of the robot is given by Equation (1). The control inputs $_L$ and $_R$ represent the linear velocities of the left and right wheel respectively. 2 denotes the distance between the wheels.

$$\begin{bmatrix} \dot{} \\ \dot{} \\ \dot{} \end{bmatrix} = \begin{bmatrix} \frac{\cos\theta}{2} & \frac{\cos\theta}{2} \\ \frac{\sin\theta}{2} & \frac{\sin\theta}{2} \\ \frac{1}{2r} & -\frac{1}{2r} \end{bmatrix} \begin{bmatrix} R \\ L \end{bmatrix} \quad (1)$$

and are related to the left and right wheel velocities $_L$ and $_R$ by the equations:

$$= \frac{_L + _R}{2} \quad ; \quad = \frac{_R - _L}{2}$$

In this paper, the following assumptions are considered:

- Obstacles are assumed to be circular.
- The positions of the robot and the obstacles can be measured accurately.
- The robot's desired position, orientation and velocity are known; with the velocity not exceeding the robot's maximum velocity.
- The robot initial position is not within an obstacle.
- The ideal target trajectory is smooth and continuous.

A. Trajectory tracking

De Luca et al.'s [7] trajectory tracking controller will be used for tracking the desired target trajectory. This consists in the introduction of a *virtual reference vehicle* to be tracked by the robot. The virtual reference vehicle defines the desired target trajectory. The controller should asymptotically reduce down to zero the coordinate error $[\ _1 \quad _2 \quad _3]^T$, detailed in Figure 2, between the real robot and the virtual vehicle. In the following, all terms with a subscript will have the same definition mentioned previously, but referred to the virtual (reference) vehicle rather than the mobile robot.

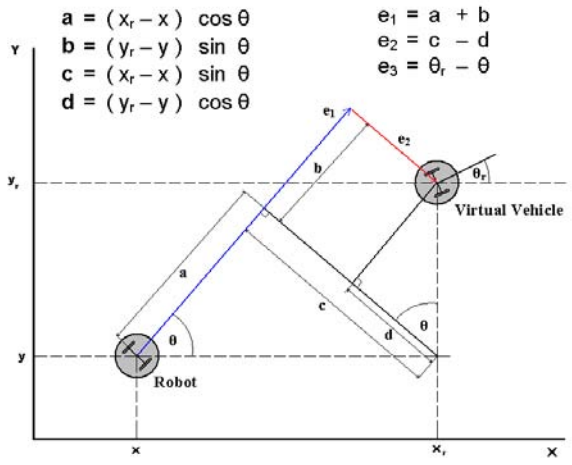

Fig. 2. Virtual vehicle following error

The desired reference co-ordinates $_r(\ )$ and $_r(\ )$ of the path traced by the virtual vehicle at any given time are provided to the controller. The calculation of the linear velocity $_r$ and angular velocity $_r$ of the virtual reference vehicle from $_r(\ )$ and $_r(\ )$ is called feedforward

command generation. Using the kinematic model Equation (1), the virtual reference vehicle orientation is given by

$$ _r = \text{atan2}(\dot{} _r \ \dot{} _r) \tag{2}$$

where atan2 is the four-quadrant inverse tangent function of $(\dot{} _r \ \dot{} _r)$, and the linear velocity is

$$ _r = \pm\sqrt{\dot{} ^2_r + \dot{} ^2_r} \tag{3}$$

The angular velocity $ _r$ is obtained by differentiating Equation (2) with respect to time yielding,

$$ _r = \frac{\ddot{} _r \dot{} _r - \ddot{} _r \dot{} _r}{\dot{} ^2_r + \dot{} ^2_r} \tag{4}$$

By applying Lyapunov stability analysis, the following control law ensuring that the robot will asymptotically track the reference vehicle was derived in [7]:

$$ = \ _r \cos(\ _r - \ ) + \ _1 [\cos \ (\ _r - \ ) + \sin \ (\ _r - \ )] \tag{5}$$

$$ = \ _r + \ _2 \ _r \frac{\sin(\ _r - \ )}{ \ _r - \ } [\cos \ (\ _r - \ ) - \sin \ (\ _r - \ )] + \ _3(\ _r - \ ) \tag{6}$$

where

$$ _1 = \ _3 = 2\sqrt{ ^2_r + \ ^2_r}$$
$$ _2 =$$

with $\ 0$ and $\ \in (0, 1)$ being design parameters.

B. Limit cycles

Nonlinear systems can display oscillations of fixed amplitude and fixed period without external excitation. These oscillations are called limit cycles [8], [9]. The amplitude of a limit cycle is independent of the initial conditions and these are not easily affected by parameter changes. Depending on the evolution of the state trajectories in the vicinity of a limit cycle, one can distinguish three kinds of limit cycles:

1) **Stable Limit Cycles:** all trajectories in the vicinity of the limit cycle converge to it as $ \to \infty$
2) **Unstable Limit Cycles:** all trajectories in the vicinity of the limit cycle diverge from it as $ \to \infty$
3) **Semi-Stable Limit Cycles:** some of the trajectories in the vicinity converge to it, while others diverge from it as $ \to \infty$

The 2nd order nonlinear system of equations given by Equation (7) was proposed for the Limit Cycle Navigation system by Kim et al. [5]. These dynamic equations result in a stable limit cycle shown in the phase portraits of Figure 3.

$$\begin{aligned} \dot{} _1 &= \ _2 + \ _1(1 - \ ^2_1 - \ ^2_2) \\ \dot{} _2 &= - \ _1 + \ _2(1 - \ ^2_1 - \ ^2_2) \end{aligned} \tag{7}$$

Figure 3(a) shows that trajectories from all initial points

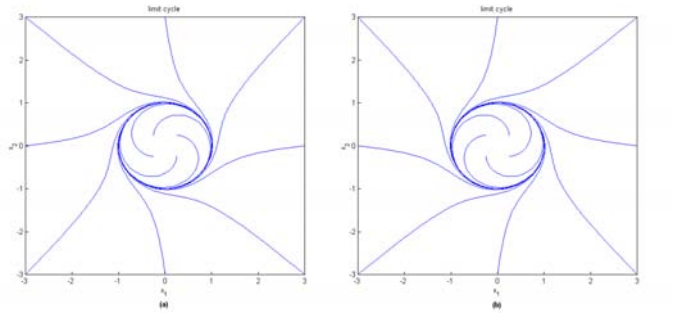

Fig. 3. (a) clockwise limit cycle, (b) counterclockwise limit cycle

$(\ _1, \ _2)$ in space move towards a circular limit cycle trajectory of radius 1, in a clockwise direction. A counter clockwise field can be easily obtained by changing the signs in Equation (7) as follows:

$$\begin{aligned} \dot{} _1 &= - \ _2 + \ _1(1 - \ ^2_1 - \ ^2_2) \\ \dot{} _2 &= \ _1 + \ _2(1 - \ ^2_1 - \ ^2_2) \end{aligned} \tag{8}$$

This yields the phase portrait of Figure 3(b). A more general form of Equation (7) will follow by introducing parameters $\ $ and $\ $ as follows:

$$\begin{aligned} \dot{} _1 &= \ \ _2 + \ _1(\ ^2 - \ ^2_1 - \ ^2_2) \\ \dot{} _2 &= - \ \ _1 + \ _2(\ ^2 - \ ^2_1 - \ ^2_2) \end{aligned} \tag{9}$$

Parameter $\ $ determines the radius of the limit cycle, whereas $\ = \pm 1$ determines the direction.

C. The Limit Cycle Navigation Method

Kim et al.'s method for Limit Cycle Navigation [5], [10] will be briefly described next. The Limit Cycle Navigation system is based on the use of the stable limit cycle dynamics defined in Equation (9) to generate a local, circular trajectory around an obstacle that needs to be avoided. This method results in an efficient solution for avoiding obstacles, which contrasts with the approach of APF methods where the robot is directed to stay as far away from obstacles as possible. Hence the Limit Cycle Navigation method does not suffer from the problem of local minima.

Consider the robot shown in Figure 4 which is asked to depart from a starting point $(6, 5)$ and head towards the goal at point$(18, 15)$. An obstacle is present in the direct line of sight between the starting point and the goal at point $(15, 12)$. Since we are considering a local, rather than a global sensing approach, the robot does not detect the obstacle immediately. This leads the robot to head directly towards the goal. As soon as the obstacle is sensed (approximately around point $(12, 10)$) the Limit Cycle Navigation system takes over. Essentially, an imaginary line $\ $ is drawn between the current robot position and the goal. The perpendicular distance $\ $ between $\ $ and the nearest obstacle

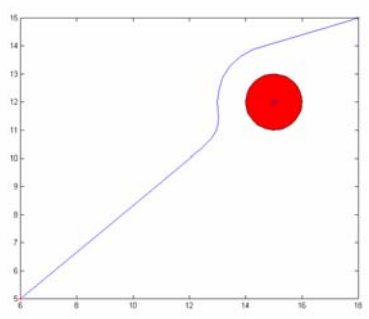

Fig. 4. Avoiding an obstacle

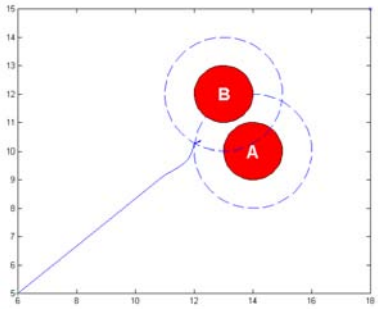

Fig. 5. Stuck between two obstacles

is then calculated using

$$= \frac{x + \quad y +}{\sqrt{\quad^2 + \quad^2}} \tag{10}$$

where(x, y),(x, y) and(x, y) are the , co-ordinates of the obstacle, the target and the robot, respectively. , and are values obtained from the straight line equation + + = 0 of line .

The local trajectory that the robot must follow to circumvent the obstacle is then given by the following equations, based upon the limit cycle dynamics of Equation (9):

$$\dot{} = \frac{}{\left|\quad\right|} + \left(\frac{^2}{v} - \quad^2 - \quad^2\right)$$
$$\dot{} = -\frac{}{\left|\quad\right|} + \left(\frac{^2}{v} - \quad^2 - \quad^2\right) \tag{11}$$

and are now interpreted as being the co-ordinates relative to the centre of the nearest obstacle considered as origin. If is positive, the robot avoids the obstacle in a clockwise manner, and vice-versa if is negative. The radius of the local trajectory, v, depends upon the size of the robot and the obstacle. Effectively,

$$v = r + o + , \tag{12}$$

where r denotes the radius of the robot and o is the radius of the obstacle. o takes a value of (the radius of a region surrounding an obstacle, called an orbit, which denotes a no-go area for the robot) when the obstacle's orbit lies on the imaginary line , hence blocking the robot's path. Alternatively, $o = 0$ when the obstacle's orbit is not obstructing the robot motion. represents a safety margin introduced as a buffer to avoid collisions. This whole process of obstacle avoidance is repeated as the robot moves and line changes with robot motion. In the example shown in Figure 4, is positive and the robot avoids the obstacle in a clockwise rotation. Once the obstacle is circumvented, the robot leaves the limit cycle orbit at a tangent and heads directly towards the goal.

The above system was also extended in [5] for obstacles having overlapping orbits. Consider the situation shown in Figure 5. The Limit Cycle Navigation method will instruct the robot to avoid obstacle by following a clockwise circle. However, the same navigation method will instruct the robot to circumvent obstacle in a counter clockwise direction. Hence, the robot will get stuck between obstacles and , in a manner similar to local minima problems in APF-based methods. To overcome this problem, Kim et al. modified the former navigation rule in the following way: If several obstacle orbits overlap, they can be regarded as one obstacle with a new equivalent central position defined by

$$x = \frac{1}{}\sum_{k=0}^{n} \quad xk \quad , \qquad y = \frac{1}{}\sum_{k=0}^{n} \quad yk \tag{13}$$

where xk and yk are the , co-ordinates of the centres of overlapping orbits. With this equivalent centre (x, y), distance for all overlapped disturbing obstacles is calculated using Equation (10). The local trajectory of the robot at each position is calculated by Equation (11) with respect to the nearest obstacle, as before.

III. Adaptation of the Limit Cycle Navigation Method for Local Sensing

The system developed by Kim et al. in [5], [10] was designed for robot soccer. In this kind of scenario, a global view of the environment is available from an overhead camera. Hence it was relatively easy to detect all the overlapping obstacles and orbits in advance, and considering these as one equivalent obstacle. Since in our case we do not have a global view of the obstacles present in the whole area of navigation, the robot navigation is based only upon local sensory information. Thus, conditions could still exist where the robot might get stuck in a loop.

Consider the situation depicted in Figure 6. Here, it is expected that the robot goes from the starting point $(6, 5)$ to the target $(18, 15)$ and the robot sensing range is limited to 3 distance units. The robot successfully circumvents the solitary obstacle . Then obstacles and are sensed. Using the previously described method due to Kim et al. [5], a new centre is calculated and is set to a negative value since the new centre will be between and . Since would be the nearest obstacle, the robot will try to avoid the obstacles in a counter clockwise direction. However, as the robot moves down towards , obstacles

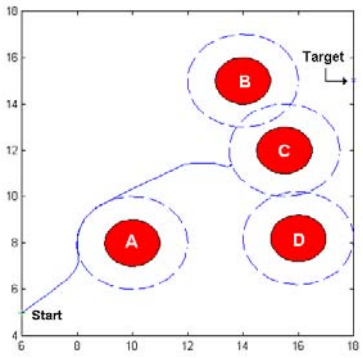

Fig. 6. Stuck in loop

and would both be detected by the local sensor. A new centre is therefore calculated and this time is set positively. This would make the robot avoid obstacle in a clockwise direction. Hence the robot would subsequently sense obstacles and , forcing the robot to rotate counter clockwise. This would lead the robot to get stuck in an undesired endless loop.

In order to avoid such situations, brought about by the constraints of local sensing, we propose to modify the navigation system of [5] by keeping the orientation of rotation fixed whilst the robot is avoiding obstacles that overlap. This would force the robot to avoid obstacles by rotating in the same direction, thus preventing the robot from getting stuck in undesired endless loops. The initial direction of rotation is chosen to be either clockwise or counter clockwise in the same way described above. However this direction is *not* changed until the robot has circumvented the obstacle that is nearest to the goal in an overlapping set. Before the robot stops using this modified Limit Cycle Navigation method to head directly towards the goal, it is to be ascertained that the distance error between the robot and the goal is reducing whilst avoiding an obstacle. Once the obstacle nearest to the goal is surpassed and the distance in error is still diminishing, the robot is allowed to leave the modified Limit Cycle Navigation system and head towards the goal. Unfortunately, it is not possible to guarantee that obstacles are avoided in the shortest possible path because of the local characteristics of the robot's obstacle sensing mechanism.

IV. Limit Cycle Navigation Combined with Trajectory Following

In our scenario the robot is required to track a desired trajectory and avoid obstacles, both static and dynamic, which might obstruct its path. This requires a combination of trajectory tracking control with obstacle avoidance navigation. When no obstacles are sensed, the robot is requested to track directly the desired trajectory by utilising control laws (5) and (6). When an obstacle or multiple obstacles are sensed, the control law is changed and the robot is asked to track the trajectory generated by the modified Limit Cycle Navigation method described in the previous section. As soon as the obstacle is circumvented, the robot switches back to the tracking control law. The algorithm being proposed in order to combine trajectory tracking with Limit Cycle obstacle avoidance is the following:

1) Specify the values of the desired trajectory to be followed *i.e.* desired position and velocities of the virtual vehicle.
2) If there are no obstacles obstructing the robot along the desired trajectory, follow it by using directly Equations (5) and (6).
3) Else, if there are obstructing obstacles
   a) Calculate distance between robot and goal
   b) Find nearest obstacle to robot
   c) Check if obstacles have overlapping orbits
   d) If yes,
      i) Calculate new centre
      ii) From the set of overlapping obstacles, find the obstacle nearest to goal
      iii) Calculate perpendicular distance between the line joining the robot and the goal, and the new obstacle centre.
   e) Else, if just one obstacle
      i) Calculate the perpendicular distance between line and the obstacle centre
   f) Use the modified Limit Cycle Navigation.
   g) Once all obstructing obstacles are surpassed, head directly towards the desired point on the desired trajectory.

The following design parameters are also required in this scheme:

1) Sensing range: This would represent the sensing range of the obstacle detection sensors being used. In our simulations, the range was set to 3 units.
2) Delay before leaving Limit cycle: The delay consists of a counter which increments each time the robot is following the limit cycle trajectory and heading towards the goal, and reset if not. If the counter value reaches some preset delay value, then the robot is heading towards the goal and is thus free to leave the limit cycle trajectory and follow the target directly.
3) Gain factor on the velocities produced by the limit cycle: The velocity determined by the limit cycle should allow for the target to remain in a slightly advanced position relative to the robot, thus giving the robot ample time to surpass the obstacle before the desired target returns to the obstacle location. This is similar to human navigation. When we find a barrier between ourselves and the goal, we allow for some time to pass to update the new co-ordinates of the goal and then devise a way to circumvent the obstacle given the new co-ordinates. Velocity is also reduced at this stage.

V. Simulation Results

Figure 7 shows the result of a simulation where the robot needs to follow a circular trajectory. A U-shaped multiple obstacle is obstructing part of this path. If the Modified APF method, as proposed in [4], were to be used for this situation, the robot gets stuck in a local minimum as shown earlier in Figure 1. The success of the approach presented in this paper is clearly shown in Figure 7 where the robot manages to circumvent the obstacle and continues to follow the virtual vehicle along the desired trajectory.

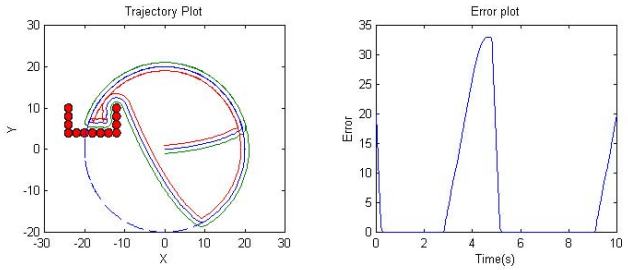

Fig. 7. Circle Trajectory: (a) robot trajectory (b) Euclidean norm of positional error

Another simulation example, this time with a figure-of-eight trajectory, is shown in Figure 8. There are two rather complex obstacles blocking this trajectory at different locations. Note that, once again, the robot successfully circumvents both obstacles. When the obstacles are surpassed, the robot tracks the desired trajectory and the error between the desired and actual position asymptotically converges down to zero over the obstacle-free regions of the path.

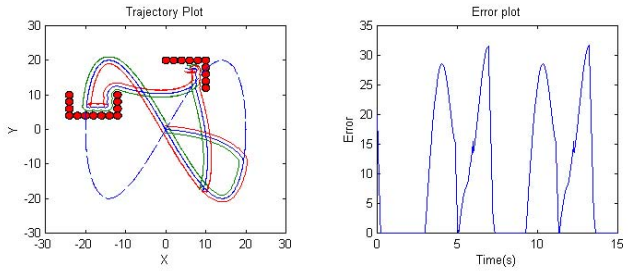

Fig. 8. Figure of eight Trajectory: (a) robot trajectory (b) Euclidean norm of positional error

VI. Conclusion and Future Directions

The nonlinear trajectory tracking control law described in Section II-A is guaranteed to drive the robot to the desired co-ordinates in the absence of obstacles [1]. In this paper we consider the case where the robot is required to follow a trajectory in the presence of obstacles. In the previous work, carried out in [4], the obstacle avoidance problem was solved by using a modified APF method. This method can suffer from problems of local minima, which is an inherent limitation of APF navigation systems.

In this paper we make use of the Limit Cycle Navigation method proposed in [5] for obstacle avoidance, so that the robot will not get stuck in potential wells. As soon as an obstacle is circumvented, the robot tracks the desired trajectory once again until another obstacle is met. Simulation results that show the effectiveness of the scheme and comparisons with APF-based methods have been presented. Additionally, the problematic implications of the Limit Cycle Navigation method when using only local obstacle sensing are discussed and modifications are proposed.

This paper has considered only the positional error between the robot and the desired trajectory. Since the system is switching between two different trajectories when obstacles are detected, a more detailed study of the robot's velocity response during a transition needs to be carried out. In addition, the dynamics of the mobile robot were not considered. If the system is to be implemented on a real mobile robot, studies on the effects of acceleration limits and the inertia of the robot will have to be considered.

The above-mentioned concerns are currently being studied and the following step would be that of implementing the system on an actual laboratory mobile robotic system to obtain experimental results and confirm the efficacy of the proposed system for industrial applications.

References

[1] G. Oriolo, A. De Luca, and M. Vendittelli, "WMR control via dynamic feedback linearization: Design, implementation, and experimental validation," *IEEE Transactions on Control Systems Technology*, vol. 10, no. 6, pp. 835–852, November 2002.

[2] O. Khatib, "Real time obstacle avoidance for manipulators and mobile robots," *The Inter. Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[3] S. Ge and Y.J.Cui, "Dynamic motion planning for mobile robots using potential field method," *Autonomous Robots 13*, pp. 207–222, 2002.

[4] R. Grech and S. G. Fabri, "Trajectory tracking of a differentially driven wheeled mobile robot in the presence of obstacles," *12th Mediterranean Conference on Control and Automation*, 2004, Turkey.

[5] D.-H. Kim, J.-H. Park, and J.-H. Kim, "Limit-cycle navigation method for soccer robot," *International Conference on Computational Intelligence, Robotics and Autonomous Systems*, 2001, Singapore.

[6] G. Dudek and M. Jenkin, *Computational Principles of Mobile Robotics*. Cambridge University Press, 2000.

[7] A. De Luca, G. Oriolo, and M. Vendittelli, *Control of Wheeled Mobile Robots: An Experimental Overview*. Springer-Verlag Heidelberg, 2001, vol. 270/2001, pp. 181–226.

[8] J.-J. E. Slotine and W. Li, *Applied Nonlinear Control*. Prentice-Hall, USA, 1991.

[9] H. K. Khalil, *Nonlinear Systems 2nd Ed.* Prentice-Hall, USA, 1996.

[10] D.-H. Kim and J.-H. Kim, "A real-time limit-cycle navigation method for fast mobile robots and its application to robot soccer," *Robotics and Autonomous Systems*, vol. 42, pp. 17–30, 2003.