Proceedings of Australasian Conference on Robotics and Automation, 2-4 Dec 2013, University of New South Wales, Sydney Australia

Decentralised Deadlock–Free Navigation for Multiple Vehicles in 2D and 3D Environments using Local Parametrised Trajectory Generation*

Michael C. Hoy and Matthew Garratt†

Abstract

Recently, Decentralised Model Predictive Control (DMPC) based approaches have been proposed for multi-vehicle vehicle guidance, and offer many advantages including robustness and allowance for vehicle dynamics. However, a method of resolving deadlocks should ideally be incorporated into these methods. In this paper, we describe a method which is able to prevent deadlocks amongst groups of vehicles. This is basically achieved by ensuring sufficient space is always present around vehicles for adjacent vehicles to be able to swap their positions when stationary. In addition, an optimisation procedure for a Parameterised Trajectory Generator (PTG) is introduced which is suitable for 3D environments, thus making the 3D path planning problem more tractable. Simulations validate the presented method.

1 Introduction

Safe, on-line navigation of multiple unmanned vehicles is an increasingly important problem in robotics. Decentralised Model Predictive Control (DMPC) has recently emerged as an ideal control framework for addressing this task. However, while many desirable properties such as cooperation and altruism have been demonstrated, deadlock avoidance is generally much harder to incorporate.

1.1 Model Predictive Control

MPC basically involves performing an on-line trajectory optimisation at every time-step, where the initial control related to the optimised trajectory is used for control. This concept can be easily extended to decentralised problems. The multi-vehicle system studied in the paper is a typical example of a networked/multi-agent control system; see e.g. Savkin and Cheng [2007]; Savkin [2004]; Matveev and Savkin [2003]; Savkin [2006] and references therein. There are five main methods that have been proposed to achieve decentralisation for MPC:

- Decentralised optimisation can find the near-optimal solution for a multi-agent system, usually by using dual decomposition, see e.g. Wakasa *et al.* [2008]; Summers and Lygeros [2012]. However this requires many communication exchanges per control update.
- Multiplexed DMPC updates the trajectory for each vehicle sequentially, when they are nearby (see e.g. Kuwata *et al.* [2007]; Siva and Maciejowski [2011]); this is similar to Sequential Decentralisation (see e.g. Adinandra *et al.* [2012]). While multiplexed MPC is suited to real time implementation, a possible disadvantage is path planning cannot occur simultaneously in two adjacent vehicles.
- Another possible solution is to require acknowledgement signals before implementing a possible trajectory, and this has the benefit of not requiring vehicles to be synchronised, see e.g. Bekris *et al.* [2012].
- Approaches also have been proposed which permit single communication exchanges per control update (see e.g. Vaccarini and Longhi [2009]; Defoort *et al.* [2009]). This is done by including a so called *coherence objective*, which essentially prevents vehicles from changing their planned trajectories significantly (after transmitting them).
- In the previous work of the authors, different types of constraints were proposed to avoid coherence objectives, though it only works with a particular type of planning algorithm (see Hoy *et al.* [2012b]).

In this paper we extend the last method, though the concepts proposed could likely be applied to other de-

*This work was supported by the Australian Research Council. Corresponding author: Michael Hoy (email: m.hoy@unsw.edu.au)

†Michael C. Hoy is with the School of Electrical Engineering and Telecommunications, the University of New South Wales, Sydney, Australia. Matthew Garratt is with the School of Engineering and Information Technology, UNSW Canberra, Australia

centralization architectures.

1.2 Planning Algorithms

Many path planning algorithms, and even some on-line MPC algorithms, have been proposed for 3D cases. However this is a much harder problem computationally, and the typical low computational capability and fast update rates required for micro-UAVs compounds this problem. Control laws with low computational complexity are often more suitable for direct implementation (see e.g. Low *et al.* [2007]; Matveev *et al.* [2011a,b]; Teimoori and Savkin [2010a,b]; Savkin and Wang [2013]; Matveev *et al.* [2012, 2011c]). Thus, it seems desirable to introduce simplified path planning algorithms for 3D cases.

Many simplified path planning algorithms have been proposed for two dimensional cases:

- The Dynamic Window selects trajectories from a class of circular trajectories (see e.g. Ogren and Leonard [2005]). The Lane Curvature Method and Beam Curvature methods are somewhat similar (see e.g. Fernandez *et al.* [2004]).
- The Parametrised Trajectory Generator (PTG) considers a larger class of trajectories (combinations of circles, spirals, asymptotes and straight lines); see e.g. Blanco *et al.* [2008]. In the previous work of the authors, maximal turns and straight segments were used and found to give good performance (see Hoy *et al.* [2012b]).

However in all these cases the trajectories are arbitrarily selected by the designer, and there is very little justification for there optimality. In addition, none of these methods have currently been extended to 3D cases, and due to their nature there is no obvious way this can be accomplished.

In this paper a PTG applicable to 3D environments is proposed, which is based on applying an optimisation to the set of trajectories that may be chosen.

1.3 Deadlock–Free Navigation

Currently deadlock free planning may be achieved using the following methods:

- If the workspace is split into a set of discrete cells, Discrete Event System (DES) based optimisation can be used to tractably avoid deadlocks. This may be decentralised, although each vehicle must essentially replicate the entire planning process (see e.g. Reveliotis and Roszkowska [2011]).
- Some local deadlock avoidance techniques called *generalised roundabout policies* have been proposed, where vehicles hold a circular holding pattern until space becomes available to advance the vehicle towards its objective – by ranking vehicles deadlocks can be avoided (see e.g. Pallottino *et al.* [2007]; Krontiris and Bekris [2011]). This paper will use a somewhat similar general concept, although we focus on applying it to MPC methods.
- Some reactive navigation approaches which maintain angular constraints between the relative velocities of pairs of vehicles has been shown to avoid deadlocks in some conditions (see e.g. Lalish and Morgansen [2012]).
- Decentralised Navigation Functions (DNF) are a class of potential field method, for which convergence to targets may be proven. However this currently uses a velocity controlled vehicle model (see e.g. Tanner and Boddu [2012]; Roussos *et al.* [2010]). Some MPC methods have been proposed which use DNFs, however the same vehicle model is used (see Roussos *et al.* [2008]). Control laws have been proposed for acceleration constrained vehicles, which are somewhat similar in nature (see Rodriguez-Seda and Spong [2012]).

A simple method for avoiding deadlocks between pairs of vehicles in environments with obstacles has also been proposed by the authors, however this does not extend easily to environments with more vehicles (see Hoy [2012]). There seems no method of preventing deadlock among arbitrary numbers of vehicles which can be easily integrated with MPC (i.e. without dividing the workspace into discrete cells).

1.4 Contribution

In this paper we propose a method of navigating groups of vehicles in both 2D and 3D, which is an extension of the decentralised collision avoidance system proposed previously by the authors in Hoy *et al.* [2012b]. This is able to provably prevent collisions between an arbitrary number of acceleration constrained vehicles with very little computational overhead. The contributions of the current paper are:

- A method of preventing deadlocks between an arbitrary number of vehicles which is easily integrated with MPC in a hierarchical control system, and does not require discrete abstractions of the workspace.
- A path planning method which generalises other local path planning approaches by applying an optimisation to a PTG.
- An implementation of DMPC which is sufficiently computationally tractable for large groups of vehicles in 3D environments.

Note proofs of theoretical statements are ommited in this version of the paper; they will be available in the full version or upon request.

2 Problem Statement

We consider a set of N acceleration constrained point vehicles operating in either a 2D or 3D workspace. For brevity we only consider holonomic vehicles free from disturbance; some other models such as unicycles and/or models with disturbance could also be used. The sampling time is assumed to be unity.

The vehicles are described by a discrete time acceleration bounded vehicle:

$$\begin{aligned} \boldsymbol{s}_i(k+1) &= \boldsymbol{s}_i(k) + \boldsymbol{v}_i(k) + \frac{1}{2}\boldsymbol{u}_i(k) \\ \boldsymbol{v}_i(k+1) &= \boldsymbol{v}_i(k) + \boldsymbol{u}_i(k), \quad \|\boldsymbol{u}_i(k)\| \leq u_{max} \end{aligned} \tag{1}$$

Here $\boldsymbol{s}_i$ is the vehicles position and equates to either $\mathbf{col}\,(s_x, s_y)$ or $\mathbf{col}\,(s_x, s_y, s_z)$ depending on whether the problem is 2D or 3D, $\boldsymbol{v}_i$ is the velocity vector, and $\boldsymbol{u}_i$ is the controlled acceleration vector. The standard Euclidean norm is used, and **col** notationally represents a column vector.

Each vehicle is associated with a stationary point target T_i and has an unique importance rank R_i which arbitrarily orders the vehicles (this could be assigned via higher order logic). A vehicle is said to reach its target when $\|\boldsymbol{s}_i - T_i\| < E$, where $E > 0$. When a vehicle reaches its target it is assumed to instantly vanish from the workspace (this is not physically realistic but simplifies analysis).

Definition 2.1 *The navigation law is successful if all vehicles must reach their targets in a bounded time, while constantly maintaining a distance of d_{safe} from each other.*

Remark 2.1 *While the control law from Hoy* et al. *[2012b] uses only one round of communication exchange per control update, in this paper, for the purpose of discrete decision making, we assume that communication can take place instantaneously and multiple times per control update with any vehicle. This is done for simplicity, and additional logic could be designed to handle communication constraints.*

3 Control Approach

The control approach basically uses passivity type arguments to ensure the distance to target of the highest ranked vehicle decreases by a finite amount within a finite time. To achieve this the vehicles are constrained to have sufficient space between them so that a swapping manoeuvre may be executed at any time.

3.1 Vehicle Separation

The bound on the minimum safe distance d_{safe} is enlarged to $d_{tar} > d_{safe}$, to allow for effects such as sampling error and disturbance. The system proposed in this paper must ensure the distance between vehicles exceeds d_{tar} at all times (including during the swapping manoeuvre).

To ensure the distribution of vehicles is sparse enough to allow adjacent vehicles to swap their positions, and additional distance d_{man} is introduced. At all times, except during the swapping manoeuvre, the distance between vehicles must exceed d_{man}.

Assumption 3.1 *It is assumed for simplicity that $d_{man} \equiv 2d_{tar}$.*

3.2 Collision Avoidance

This section summarises necessary information from Hoy *et al.* [2012b]. The reader should refer to that paper for additional details. The concept consists of each vehicle planning *probational* trajectories at each time instant, which terminate with the vehicle stationary. As an outline, the algorithm to achieve a separation of d_{man} consists of sequential execution of the following steps:

S.1 Generation of a finite set $\mathcal{P}$ of planned trajectories, each starting at the current robot state $\boldsymbol{s}(k), \boldsymbol{v}(k)$; this set is chosen based on some heuristic for trajectory candidates.

S.2 Refinement of $\mathcal{P}$ to only mutually feasible trajectories; specifically, we discount the trajectories that interfere with:

r.1) Probational trajectories of the other vehicle, received from them within the time interval $[k-1, k]$, such that:

$$\|\boldsymbol{s}^*(j|k) - \boldsymbol{s}_i(j+1|k-1)\| < d_{man} \tag{2}$$

r.2) Any *presumable planned trajectory* of the other vehicle at the previous time step $k-1$. These are the trajectories that don't interfere with the trajectory transmitted from the vehicle at hand at $k-1$, and are denoted with the hat $\widehat{\{\}}^*$:

$$\begin{aligned} &\|\boldsymbol{s}^*(j|k) - \widehat{\boldsymbol{s}}_i^*(j|k)\| > d_{man} \\ &\forall \widehat{\boldsymbol{s}}_i^*(j|k) : \|\widehat{\boldsymbol{s}}_i^*(j|k) - \boldsymbol{s}(j+1|k-1)\| > d_{man} \end{aligned} \tag{3}$$

.

S.3 Selection of a trajectory:

- If $\mathcal{P}$ is empty, the probational trajectory is inherited from the previous time step (modulo a proper time advancement).
- If $\mathcal{P}$ is not empty, a trajectory is taken from $\mathcal{P}$ by minimising some cost functional.

S.4 Application of the first control $\boldsymbol{u}(0|k)$ related to the selected probational trajectory to the robot.

S.5 Transmission of the chosen probational trajectory and the current state $\boldsymbol{s}(k), \boldsymbol{v}(k)$ to the other vehicles.

S.6 $k := k + 1$ and go to S.1.

Here the notation $(\cdot)^*$ denotes probational trajectories which have not yet been selected; it is dropped when it has been chosen and implemented. The indexing $(j|k)$ represents the time-step $j + k$ when the current time is k. Variables subscripted by i refer to other vehicles, the absence of this subscript indicates the variable is associated with the vehicle at hand.

3.3 Trajectory Selection

A possible vehicle position is termed *convergent* if the distance to target decreases by at least $d_{dec} > 0$. For any chosen trajectory, if there are no higher ranked vehicle(s) in communication radius, we require that all points on the trajectory are convergent:

$$\max_{j>0} \|\boldsymbol{s}^*(j|k) - T\| < \|\boldsymbol{s} - T\| - d_{dec} \tag{4}$$

Here $d_{dec} > 0$ is a given decrement.

Out of the trajectories satisfying this constraint, the best trajectory was selected simply to minimise the distance to the target, from any point on the trajectory. It should be emphasised this is purely arbitrary, and was found to give adequate performance for our application. Other possibilities include lower ranked vehicles attempting to ensure higher ranked vehicles always have a viable trajectory, however this seems to be more important when static obstacles are also present in the environment (see e.g. Hoy [2012]).

Assumption 3.2 *The distances E and d_{dec} are constrained so:*

$$E > \frac{(d_{dec} + \frac{1}{2}d_{man})^2 - d_{man}^2}{2d_{dec}}$$

3.4 Swapping Manoeuvre

In this approach, deadlock avoidance is achieved by including a low level behaviour which allows two nearby adjacent vehicles to swap their positions. Intuitively, if a higher ranked vehicle can always swap positions with another vehicle and progress towards its goal, deadlocks may be avoided. The only requirement is that vehicles are sufficiently spaced to allow such transitions.

When the highest ranked vehicle is stationary, it executes the following steps:

1. The vehicle at hand verifies it cannot plan a trajectory using the normal trajectory planner at the current time step.
2. All vehicles with any point on their probational trajectory within $2d_{man}$ of the vehicle at hand are instructed to either stop or re-plan a trajectory respecting these bounds.
3. The vehicle waits for the vehicles from step 2) to halt. If at any time during this manoeuvrer the vehicle at hand can plan a trajectory decreasing the distance to target, it executes this trajectory and releases the constraints on the other vehicles.
4. A vehicle is selected, with which the swapping manoeuvre is performed. This is chosen as the vehicle within a distance $(2d_{man} - \epsilon)$ which maximises the decreases in the distance to target for the vehicle at hand, and for which the line segment connecting the vehicles does not pass with d_{tar} of any other vehicle (ϵ excludes vehicles at the exact distance $2d_{man}$). If no such vehicle exists, a 'virtual vehicle' is chosen instead (see Fig. 2). This is the point obtained by moving exactly d_{man} towards the target from the vehicle's current position.
5. The swapping manoeuvrer begins (all other vehicles remain stationary).
6. Once the swapping manoeuvre is complete, normal operation ensues.

The swapping manoeuvre itself is precisely defined through the following steps. These refer to Fig. 1, which indicates the manoeuvre for the 2D case. The 3D case is very similar; the manoeuvre takes place on a two dimensional plane taken chosen arbitrarily from the space. It should be reiterated that this is only one possible swapping manoeuvre selected for ease of description, more efficient manoeuvres could certainly be devised.

1. A primary axis is constructed passing through the two vehicles. An orthogonal secondary axis is also constructed (in the 3D case its choice is taken arbitrarily from the orthogonal plane).
2. The points (**A**), (**B**) and (**C**) are constructed by moving a distance $\frac{1}{\sqrt{2}}d_{tar}$ along these two axis according to Fig. 1.
3. The higher ranked vehicle (**1**) moves to (**A**), while the lower ranked vehicle (**2**) moves to (**B**).
4. (**2**) moves to (**C**).
5. (**1**) moves to (**B**).
6. The vehicles (**1**) and (**2**) move to their respective final positions.

The trajectory generated to move between arbitrary points is described as follows. Given a particular target point $\boldsymbol{s}_{tar}$, assuming the vehicle is stationary:

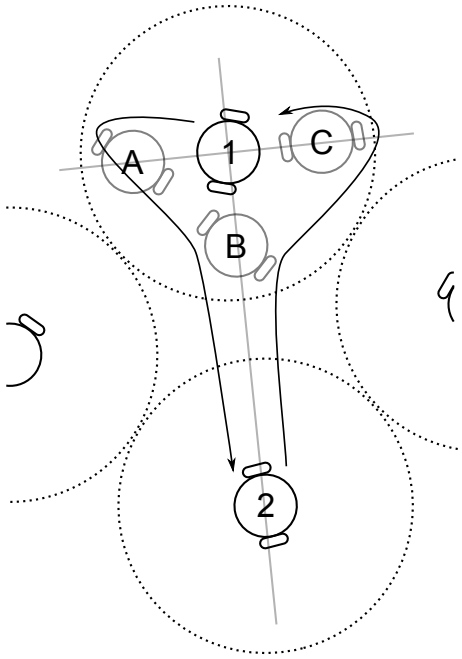

Figure 1: Provided there is sufficient space around vehicles, adjacent vehicles are always able to carry out a swapping maneuver. The points A, B, and C are intermediate positions of the vehicles.

$$\boldsymbol{u}^*(j|k) := \frac{\boldsymbol{s}_{tar} - \boldsymbol{s}(k)}{k_c^2} \cdot \begin{cases} +1 & j < k_c \\ -1 & j \geq k_c \end{cases}$$
$$\text{where} \quad k_c := \left\lceil \sqrt{\frac{\|\boldsymbol{s}_{tar} - \boldsymbol{s}\|}{u_{max}}} \right\rceil \quad (5)$$

Here $\lceil \cdot \rceil$ is the integer ceiling function. This process is summarised by the scenario diagram in Fig. 1. Note each step may be completed in finite time.

Remark 3.1 *In this description it was specified that only the highest ranked vehicle may perform a swapping manoeuvre. However this was done for simplicity, and in fact extended logic would allow the locally highest ranked vehicle to perform the swapping manoeuvre.*

4 Off-line Optimisation of the Parametrised Trajectory Generator

Trajectory generation could potentially take many forms, however for the simulations presented in this paper we choose trajectories from a set of possibilities computed off-line. Each trajectory is exactly N_p time-steps in length, and the vehicle is required to be stationary at the termination of the trajectory. Specifically, the trajectory planner can be prescribed by the control $\boldsymbol{u}^*(j|k)$ to be applied to the vehicle at each time step.

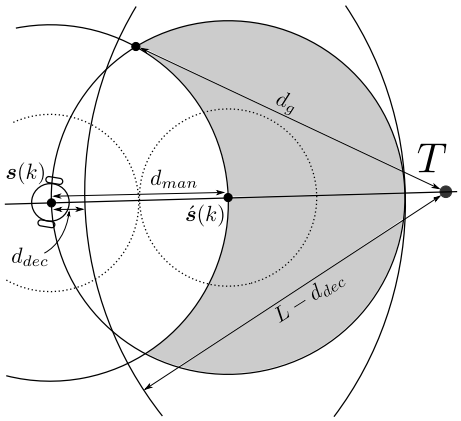

Figure 2: If a vehicle cannot be found to perform the swapping manoeuvre with, a 'virtual vehicle' is chosen instead. The swapping manoeuvre proceeds as normal.

There are N_t candidate trajectory profiles of length N_p, leading to a total of $(N_t \cdot N_p \cdot 3)$ coefficients being required to prescribe the trajectory planner. For this algorithm, the planner would look at most N_p time steps into the future. The coefficients can be optimised off-line using a global optimisation approach, such as the genetic algorithm.

To ensure the vehicle is stationary at the end of the trajectory, the final section of the trajectory is used to slow the vehicle down. This part of the trajectory planner specification was not optimise. The number of time-steps required to halt the vehicle is dependent on the vehicle's speed; if $\|\boldsymbol{v}^*(N_p|k)\| < u_{max}$, then $\boldsymbol{u}^*(N_p|k) := -\boldsymbol{v}^*(N_p|k)$. Else if $\|\boldsymbol{v}^*((N_p-1)|k)\| < 2u_{max}$, then $\boldsymbol{u}^*((N_p-1)|k) := \boldsymbol{u}^*(N_p|k) := -\frac{1}{2}\boldsymbol{v}^*((N_p-1)|k)$. This process is continued as necessary on subsequent time steps of the trajectory.

To choose the vector of coefficients, the Genetic Algorithm and Direct Search Toolbox 7.7.0.471 from MATLAB 2008b was used. The quality of the vector was evaluated by performing the simulation shown in Sec. 6 with the nominal vehicle model (2), followed by finding the sum of the time taken for each vehicle to be within $2m$ of their respective targets.

The selected trajectories are summarised by Fig. 3, and the final position perturbations obtained after applying each trajectory to a stationary vehicle are shown in Fig. 4.

Remark 4.1 *It is difficult to compare the performance of the trajectory planner since no other PTG have been previously proposed for 3D environments. In the case of a simple 2D environment, a preliminary comparison was made with alternatives from Hoy* et al. *[2012a] and Blanco* et al. *[2008], and similar performance was observed. However our approach requires less "guesswork",*

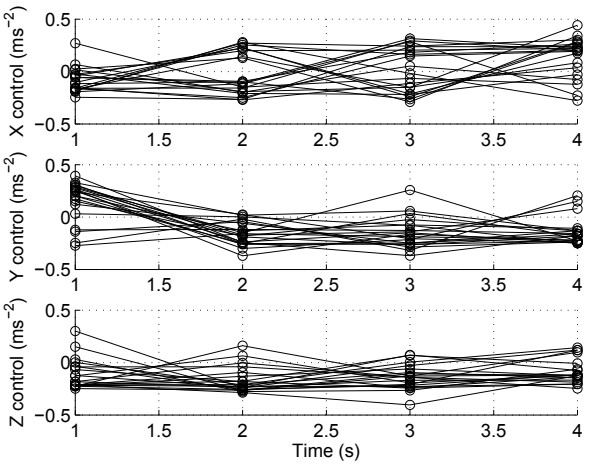

Figure 3: Controls related to the chosen trajectories selected by the GA optimisation.

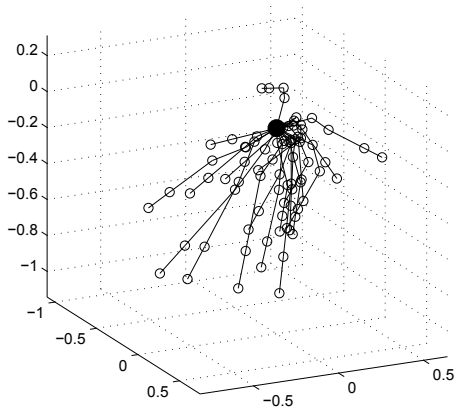

Figure 4: The set of possible final positions corresponding to applying each possible trajectory to a stationary vehicle.

and thus may be easier to implement when the trajectories are limited to a given number. Also there is no simple, obvious way to extend the previously mentioned trajectories to 3D environments.

5 Analysis

In this section we demonstrate the correct behaviour of the navigation system. Note proofs of theoretical statements are ommited in this version of the paper.

Lemma 5.1 *If the highest ranked vehicle is stationary and there are no other non-stationary vehicles with trajectory points within a distance* $2d_{man}$*, either there is a vehicle within the distance* $(2d_{man} - \epsilon)$ *whose position is convergent, and the line segment connecting them is further than* d_{tar} *from any other vehicle; or there exists a feasible position corresponding to a virtual vehicle; or the trajectory planner produces a convergent trajectory.*

Lemma 5.2 *The swapping manoeuvre is collision free.*

Theorem 5.1 *All vehicles converge to their targets.*

6 Simulations

Initial simulations were carried out with the nominal vehicle model. For all these simulations, the controller sampling time was $0.5s$, u_{max} was $4.0ms^{-2}$ and d_{tar} was $4m$. It can be seen all vehicles converged towards their respective targets while avoiding collisions. The closed loop trajectory, distance time history, control time history and velocity time history are shown in Figs. 5 to 8 respectively.

The average computation time of the control law (not including overhead such as communication and simulation of the vehicle model) in this scenario was 1.7 ms with a standard deviation of 2.1 ms, which was achieved with a C++ mex function (all other parts of the simulation were implemented in MATLAB) executed on a 2.3 GHz Intel Core i7 processor. The swapping manoeuvre was unable to be tested as conditions which required its use were unable to be found, thus these simulations are mainly validating the proposed PTG.

Note that there were small increases in the target distance later in the simulation, however these occurred after the point where the respective vehicle was assumed to disappear from the workspace in Sec. 2. Also note the steps in the control signal occur because of their relatively slow update rate. This relatively small computation time makes the method suitable for real time control, and a similar method was successfully implemented on mobile robots in Hoy *et al.* [2012b].

7 Conclusions

In this paper a method of preventing deadlocks between groups of vehicles is described, which is suitable for 2D and 3D environments without static obstacles. This method seems more suitable for decentralised model predictive control than existing methods. In addition to this, a method of optimising a Parametrised Trajectory Generator (PTG) is described.

Future work will look at validation with higher fidelity vehicle models, manoeuvres which are more efficient than the swapping manoeuvre described in this paper, and also alternate path optimisation procedures.

8 References

S. Adinandra, E. Schreurs, and H. Nijmeijer. A practical model predictive control for a group of unicycle mobile robots. In *Proceedings of the 4th IFAC Conference on*

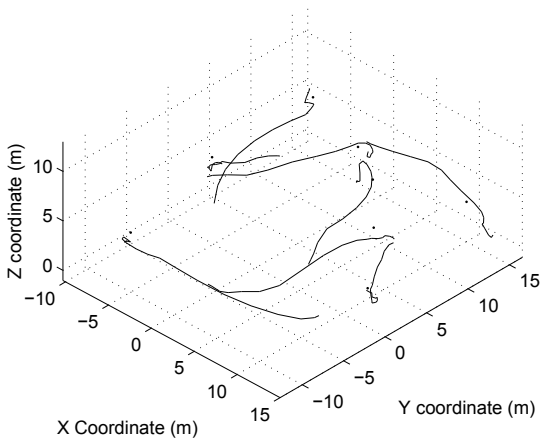

Figure 5: Closed loop Trajectories for the multivehicle system.

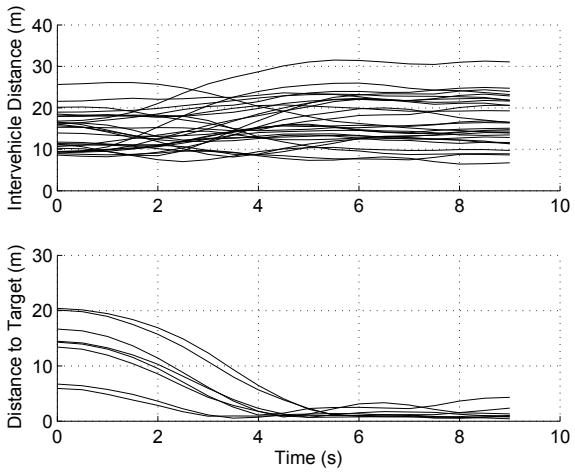

Figure 6: Inter-vehicle and target distances.

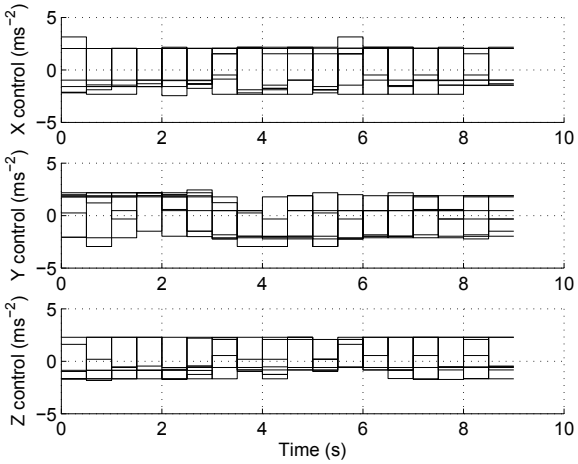

Figure 7: Control time history.

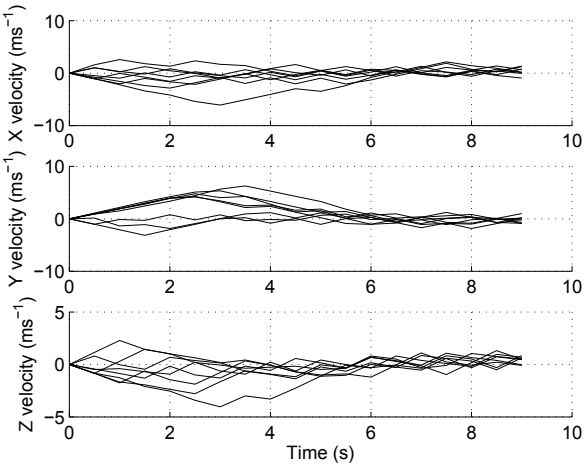

Figure 8: Velocity time history.

Nonlinear Model Predictive Control, volume 4, pages 472–477, Leeuwenhorst, Netherlands, 2012.

K. E. Bekris, D. K. Grady, M. Moll, and L. E. Kavraki. Safe distributed motion coordination for second–order systems with different planning cycles. *The International Journal of Robotics Research*, 31(2):129–150, 2012.

J-L. Blanco, J. Gonzalez, and J-A. Fernandez-Madrigal. Extending obstacle avoidance methods through multiple parameter–space transformations. *Autonomous Robots*, 24(1):29–48, 2008.

M. Defoort, A. Kokosy, T. Floquet, W. Perruquetti, and J. Palos. Motion planning for cooperative unicycle–type mobile robots with limited sensing ranges: A distributed receding horizon approach. *Robotics and Autonomous Systems*, 57(11):1094–1106, 2009.

J. L. Fernandez, R. Sanz, J. A. Benayas, and A. R. Diaguez. Improving collision avoidance for mobile robots in partially known environments: the beam curvature method. *Robotics and Autonomous Systems*, 46(4):205–219, 2004.

M. Hoy, A. S. Matveev, M. Garratt, and A. V. Savkin. Collision free navigation of an autonomous unmanned helicopter in unknown urban environments: Sliding mode and MPC approaches. *Robotica*, 30(4):537–550, 2012.

M. Hoy, A. S. Matveev, and A. V. Savkin. Collision free cooperative navigation of multiple wheeled robots in unknown cluttered environments. *Robotics and Autonomous Systems*, 60(10):1253–1266, 2012.

M. Hoy. Deadlock resolution for navigation of wheeled robots in continuous state-space. In *Proceedings of the International Conference on Automation, Robotics, Control and Vision*, Guangzhou, China, 2012.

A. Krontiris and K. E. Bekris. Using minimal communication to improve decentralized conflict resolution for non-holonomic vehicles. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, pages 3235–3240, San Francisco, CA, 2011.

Y. Kuwata, A. Richards, T. Schouwenaars, and J. P. How. Distributed robust receding horizon control for multi-vehicle guidance. *IEEE Transactions on Control Systems Technology*, 15(4):627–641, 2007.

E. Lalish and K. Morgansen. Distributed reactive collision avoidance. *Autonomous Robots*, 32(3):207–226, 2012.

E. M. Low, I. R. Manchester, and A. V. Savkin. A biologically inspired method for vision–based docking of wheeled mobile robots. *Robotics and Autonomous Systems*, 55(10):769–784, 2007.

A. S. Matveev and A. V. Savkin. The problem of state estimation via asynchronous communication channels with irregular transmission times. *IEEE Transactions on Automatic Control*, 48(4):670–676, 2003.

A. S. Matveev, H. Teimoori, and A. V. Savkin. A method for guidance and control of an autonomous vehicle in problems of border patrolling and obstacle avoidance. *Automatica*, 47(3):515–514, 2011.

A. S. Matveev, H. Teimoori, and A. V. Savkin. Navigation of a unicycle–like mobile robot for environmental extremum seeking. *Automatica*, 47(1):85–91, 2011.

A. S. Matveev, H. Teimoori, and A. V. Savkin. Range–only measurements based target following for wheeled mobile robots. *Automatica*, 47(1):177–184, 2011.

A. S. Matveev, C. Wang, and A. V. Savkin. Real–time navigation of mobile robots in problems of border patrolling and avoiding collisions with moving and deforming obstacles. *Robotics and Autonomous Systems*, 60(6):769–788, 2012.

P. Ogren and N. E. Leonard. A convergent dynamic window approach to obstacle avoidance. *IEEE Transactions on Robotics*, 21(2):188–195, 2005.

L. Pallottino, V. G. Scordio, A. Bicchi, and E. Frazzoli. Decentralized cooperative policy for conflict resolution in multi-vehicle systems. *IEEE Transactions on Robotics*, 23(6):1170–1183, 2007.

S. A. Reveliotis and E. Roszkowska. Conflict resolution in free–ranging multi-vehicle systems: A resource allocation paradigm. *IEEE Transactions on Robotics*, 27(2):283–296, 2011.

E. J. Rodriguez-Seda and M. W. Spong. Guaranteed safe motion of multiple lagrangian systems with limited actuation. In *Proceedings of the 51st IEEE Conference on Decision and Control*, pages 2773–2780, Maui, HI, 2012.

G. P. Roussos, G. Chaloulos, K. J. Kyriakopoulos, and J. Lygeros. Control of multiple non-holonomic air vehicles under wind uncertainty using model predictive control and decentralized navigation functions. In *Proceedings of the 47th IEEE Conference on Decision and Control*, pages 1225–1230, Cancun, Mexico, 2008.

G. Roussos, D. V. Dimarogonas, and K. J. Kyriakopoulos. 3D navigation and collision avoidance for nonholonomic aircraft-like vehicles. *International Journal of Adaptive Control and Signal Processing*, 24(10):900–920, 2010.

A. V. Savkin and T. M. Cheng. Detectability and output feedback stabilizability of nonlinear networked control systems. *IEEE Transactions on Automatic Control*, 52(4):730–735, 2007.

A. V. Savkin and C. Wang. A simple biologically-inspired algorithm for collision free navigation of a unicycle-like robot in dynamic environments with moving obstacles. *Robotica*, 31(6):993–1001, 2013.

A. V. Savkin. Coordinated collective motion of groups of autonomous mobile robots: Analysis of Vicsek's model. *IEEE Transactions on Automatic Control*, 49(6):981–983, 2004.

A. V. Savkin. Analysis and synthesis of networked control systems: Topological entropy, observability, robustness, and optimal control. *Automatica*, 42(1):51–62, 2006.

E. Siva and J. M. Maciejowski. Robust multiplexed MPC for distributed multi–agent systems. In *Proceedings of the 18th IFAC World Congress*, Milano, Italy, 2011.

T. H. Summers and J. Lygeros. Distributed model predictive consensus via the alternating direction method of multipliers. In *Proceedings of the Allerton Conference on Communication, Control, and Computing*, Monticello, IL, 2012.

H. G. Tanner and A. Boddu. Multiagent navigation functions revisited. *IEEE Transactions on Robotics*, 28(6), 2012.

H. Teimoori and A. V. Savkin. A biologically inspired method for robot navigation in a cluttered environment. *Robotica*, 28(5):637–648, 2010.

H. Teimoori and A. V. Savkin. Equiangular navigation and guidance of a wheeled mobile robot based on range–only measurements. *Robotics and Autonomous Systems*, 58(2):203–215, 2010.

M. Vaccarini and S. Longhi. Formation control of marine vehicles via real–time networked decentralized MPC. In *Proceedings of the 17th Mediterranean Conference on Control and Automation*, pages 428–433, Thessaloniki, Greece, 2009.

Y. Wakasa, M. Arakawa, K. Tanaka, and T. Akashi. Decentralized model predictive control via dual decomposition. In *Proceedings of the 47th IEEE Conference on Decision and Control*, pages 381–386, Cancun, Mexico, 2008.