2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

Integrated Planning and Control of Large Tracked Vehicles in Open Terrain

Xiuyi Fan[1], Surya Singh[2], Florian Oppolzer[3]
Eric Nettleton[2], Ross Hennessy[2], Alexander Lowe[2], and Hugh Durrant-Whyte[2]
[1]Department of Computing, Imperial College London, United Kingdom
[2]Australian Centre for Field Robotics, University of Sydney, Australia
[3]Technology and Innovation, Rio Tinto, Australia
x.fan09@imperial.ac.uk, florian.oppolzer@riotinto.com, {spns, e.nettleton, r.hennessy, a.lowe, hugh}@acfr.usyd.edu.au

Abstract— **Trajectory generation and control of large equipment in open field environments involves systematically and robustly operating in uncertain and dynamic terrain. This paper presents an integrated motion planning and control system for tracked vehicles. Flexible path-end adjustments and adaptive look-ahead are introduced to a state lattice planning approach with waypoint control. For a given processing horizon, this increases search coverage and reduces planning error.**

This tramming approach has been successfully fielded on a 98-ton autonomous blast hole drill rig used in iron ore mining in Western Australia. The system has undergone extensive testing and is now integrated into a production environment. This work is a key element in a larger program aimed at developing a fully autonomous, remotely operated mine.

I. Introduction

Due to the remoteness and potential hazards in mining, automation is desirable to achieve higher efficiency and safety [1]. One of the vehicles central to open-pit mining operations is a surface drilling rig (Fig. 1). This large tracked vehicle drills holes into the ground that are subsequently filled with explosive and blasted. The fragmented rock is then extracted. Drill motions are not arbitrary. Hole locations are carefully planned in regular patterns according to the underlying geology. Thus, there is an underlying structure in the form of preferred maneuvers, such as straight-line tramming, row shifting, or three-point turning.

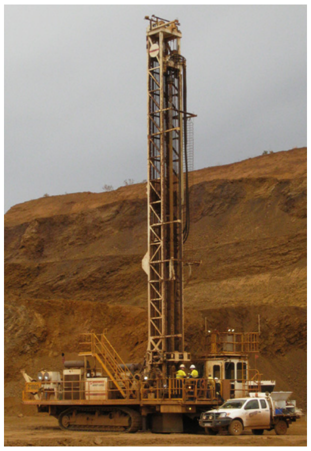


Fig. 1. The autonomous blast hole drill rig at rest on a drill bench of an open-pit mine (the 4x4 truck illustrates the large scale).

Drill rig automation can be divided into three modules: tramming, levelling, drilling. This paper focuses on tramming, i.e., the processes of travelling between locations. A multi-goal planning and control problem, tramming can be separated into three main tasks: 1) determining the appropriate location sequence given a pattern; 2) calculating trajectories between initial and final pairs of drill locations; and 3) driving the drill to follow these trajectories.

In the fielded implementation, ordering is specified *a priori* or can be determined by approximate methods (detailed in related work [2]). The paths are formulated using state lattices; and control is performed via a feed-forward, pure-pursuit law. State lattices are a path set approach and operates in a discrete-space, discrete-time manner with reachable sets of control sequences tessellated through space at set time intervals [3]. By reducing the problem from a continuum of motions to a search through a discrete lattice, this provides rapid computation and assures traversability. However, it is only complete up to the sampled control resolution. In comparison to probabilistically complete sampling methods (e.g., RRTs), it provides fixed-time operation, efficiency for low dimensional problems, and a direct means for exploiting the structure through preferred maneuvers.

The operating environment adds complications that preclude direct application. The vibration resulting from the drill's suspension compliance, large forces, friction, and inertia make control difficult. In particular, for greater satellite visibility in deep pits, the GPS antennas rest atop the drill mast. The consequence of which is that locomotive impulses lead to mast sway of $\sim 30\ cm$, which without control causes positioning errors. Computation is limited because of the harshness of the environment. Finally, actions from a set are better since the drill may be operated in a supervisory mode.

The paper introduces an adaptive look-ahead and a flexible path-end adjustment modification scheme. This provides efficient planning with sufficient operating coverage to yield traversable, controllable, and obstacle-robust paths. Tracking achieves positioning accuracy in the presence of sensor uncertainly on par or better than manual operation. This paper starts with the software architecture and then details the modified state lattices planner. Later sections introduce performance metrics and detail field operation.

II. Architecture Overview

Considerable research in the tracked vehicle domain has been focused on model-based operations with explicit kinematic and terrain models, particularly in regards to track-ground interaction [4]. Approaches range from identification of track slippage without previous knowledge of soil characteristics [5] to straight line motion control that regulates as error the mechanical asymmetries of the tracks [6]. This has been extended to kinetic approaches that calculate the required track forces to follow a path at given speeds [7]. In the mining vehicle context, a tracked vehicle controller inspired by wheeled robots is introduced and simulated in [8]. Whereas in [9], two separate controllers are shown to realize controlled translational and angular velocities.

As the overarching goals of the autonomous tramming system are to safely and robustly navigate the drill through drill-hole patterns, it is desirable to perform navigation without previous knowledge of soil characteristics and to avoid obstacles. This entails a number of sub-requirements in addition to the determination of steering commands. Chiefly the autonomous tramming system needs: (1) to determine a traversable path that covers all holes in a pattern while maintaining tight spatial constraints; and (2) to follow the planned path precisely and stop at holes accurately. These tasks may be intuitive to an experienced operator, but its principled determination is non-trivial.

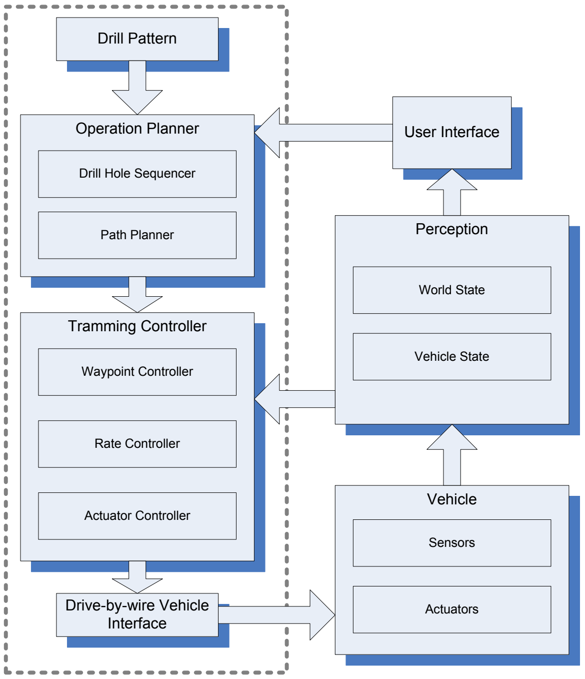


Fig. 2. Overview of the drill system architecture. The dashed line indicates planning and control subsystems

Figure 2 gives a simplified view of the system architecture. The entire system can be viewed as a loop of messages. The operation planner receives drill-hole patterns and user inputs, which are various parameter settings, and generates vehicle path plans. The two sub-components of the operation planner are the drill-hole sequencer and path planner. The drill-hole sequencer (detailed in [2]) is responsible for generating the drilling order for all holes in a pattern, whereas the path planner is responsible for computing traversable paths that connect all holes. Path plans are then fed to the tramming controller module, which is responsible for executing a plan. The outputs of the tramming controller, which are actuator control commands, are sent to a drive-by-wire hardware interface and then subsequently executed by the hardware actuators. At the same time, the vehicle constantly collects information about its environment and internal states, and processes and sends this information to both of the tramming controller and the user interface. This information is utilized by the tramming controller to control the drill and the human user to monitor the drill progress.

III. State Lattice Path Planner

A state lattice path planning approach [3] is adopted as the environment has not overly cluttered and the system has three degrees of freedom. A search graph is composed of vehicle configuration as nodes and primitive trajectories (formed from forward integrating discrete sets of vehicle commands) as links (see Table I and Fig. 3). Minimal (not necessarily optimal) paths are found by searching the graph. Furthermore, since the machine is non-holonomic, the heuristic function (for A* search) is nontrivial. An interpolation table approach [10] is adopted based on non-holonomic distance metrics [11].

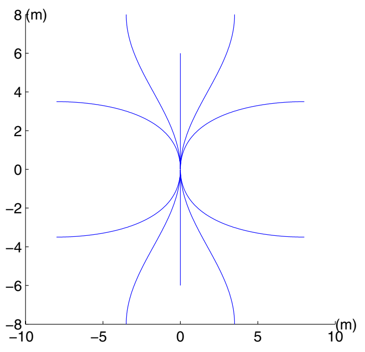
TABLE I
End Nodes of the Ten Primitive Trajectories.

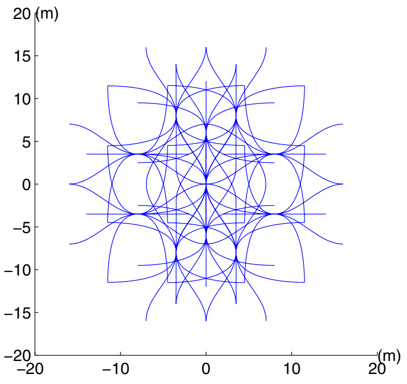
Node	Northing	Easting	Heading
1	6	0	0
2	-6	0	0
3	8	3.5	0
4	8	-3.5	0
5	-8	3.5	0
6	-8	-3.5	0
7	3.5	8	$\pi/2$
8	3.5	-8	$-\pi/2$
9	-3.5	8	$-\pi/2$
10	-3.5	-8	$\pi/2$

Performance is improved by defining the primitive trajectories based on a functional study of tramming trajectories in production operation. In contrast to standard implementations, in which the primitive trajectories are defined by discretizing the reachable configurations based on the vehicle kinematics, this approach is able to exploits histories from both expert drivers and previous autonomous operations.

This results in a set of trajectories that is a subset of all reachable configurations, as unlike wheeled vehicles, which have their maneuverability limited by vehicle dynamics, the dominant factor for drill trajectory regulation comes from terrain constraints and drill operation procedures.

In addition to defining primitive trajectories according to drill operations, we employ a more flexible approach for trajectory generation. One inherited limitation of the standard state lattices approach is the dilemma of the search branching factor and the coverage of the search space. Once the vehicle configuration discritisation is involved, there is


(a) Ten primitive trajectories (*zero* rounds of node expansion)


(b) Trajectories after *one* round of node expansion

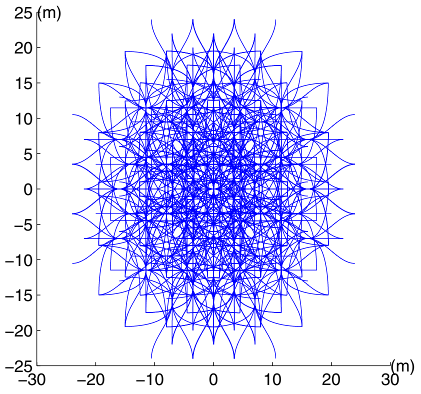

(c) Trajectories after *two* rounds of node expansion

Fig. 3. The ten primitive trajectories and their first and second expansions

a compromise between the number of primitive trajectories and the coverage of the vehicle configuration space, i.e., the fewer primitive trajectories, the faster the search. However, there are more unreachable configurations in such a system. To ensure sufficient coverage, more primitive trajectories are required. This results in a higher computational burden to evaluate all possible path compositions.

The core innovation of our approach is centered at the flexibility we introduce at the primitive trajectory generation. We generate ten primitive trajectories represented using cubic Bezier splines. However, instead of generating primitive trajectory compositions until the path is found, we first generate a cubic spline path from the current search node to the final destination. In the case of an invalid path, the current search node is expanded with the ten pre-defined primitive splines. This strategy ensures that there is no vehicle configuration error incurred at the path planning stage due to the search space discritisation. Given a set of primitive trajectories that is complete enough, this strategy also fills all holes created by the discritisation.

Given the relatively short path lengths, the final vehicle configuration can be reached within three or four levels of node expansion. There is no need to design a sophisticated heuristic function to guide the search. Instead, breadth-first search is used. The algorithm stops when a set of paths is found or it reaches a node expansion threshold. We then evaluate all resulting paths based on criteria such as the path length and the amount of turning. The path ranks at the top in such evaluation is returned.

Further, the implementation composes trajectories of both a cubic Bezier spline *and* a straight line segment (see Fig. 4) The purpose of the straight segment ending is to ease the trajectory tracking problem for the vehicle controller. This is also in a framework that is similar to and compatible with optimal steering methods [12].

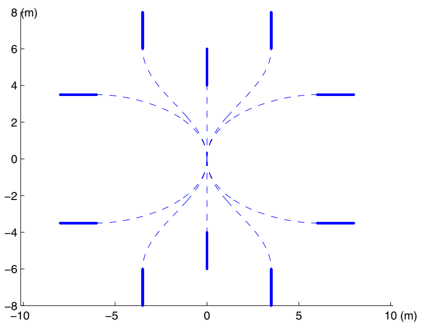


Fig. 4. Ten primitive trajectories with straight-line segment ending

Since trajectories are generated using splines, they are represented by discritised points. It is very convenient to compute the amount of heading change between two adjacent points. Since there is a direct mapping between the heading change and the turning radius, we use this measure to evaluate the amount of turning for each trajectory, instead of the curvature.

IV. Tramming Controller

The tramming controller is composed of three sub-controllers in a hierarchical manner (see also Fig. 5). It is designed such that the actuator controller regulates the speed of the two tracks; the rate controller regulates the vehicle velocity and the turn rate; and the waypoint controller maintains the tracking of planned trajectories. Due to the skid steering nature of the drill, velocity and yaw rate are chosen as the two main control parameters because they directly relate to the drills tramming commands.

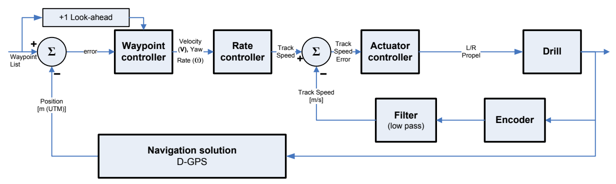


Fig. 5. Block diagram of the tramming controller showing the control sequences and feedback for an implementation of the control system on a blast-hole drill rig. Input shaping can be optionally used to actively control vibration modes.

A. *Tramming Controller Design Requirement*

Unlike autonomous vehicles designed for general purpose road driving, autonomous drill tramming has a specific set of requirements. In contrast to road vehicles that operate in a highly dynamic environment, the working environment for drill is static. On-site traffic is tightly monitored and controlled. Moving obstacles in the field are rare. Therefore, there is no requirement for high speed traversing and agile maneuverability. On the other hand, due to the tight spatial constrains on drill sites and to ensure the accurate positioning of drill holes, drill tramming requires considerably higher control accuracy than road vehicles. This requirement translates to tight error tolerances on path tracking and final vehicle positioning.

B. *Actuator Controller*

The actuator controller is the logic nearest to the track actuators. It takes the speed and yaw rate as its inputs from the rate controller and generates track speeds as its outputs.

Assuming the control point of the drill is at its center of rotation, using a clockwise positive coordinate frame, we have:

$$\dot{\theta} = \frac{SR - SL}{Base} \tag{1}$$

$$v = \frac{SR + SL}{2} \tag{2}$$

where SR is the speed of the right track, SL is the speed of the left track, and Base is the distance between the two tracks. Solving for SR and SL gives:

$$SR = \frac{2v + Base\dot{\theta}}{2} \tag{3}$$

$$SL = \frac{2v - Base\dot{\theta}}{2} \tag{4}$$

We use a PID controller to control these two track speeds with a pressure relief for safety. Feedback is taken from the two track encoders.

C. *Waypoint Controller*

The waypoint controller is the top level tracking controller in this drill tramming system. It takes waypoint lists, which are coordinates, as its inputs from the path planner and generates speed and yaw rate. The trajectory tracking control is inspired by various sources, which include the Argo [13] and the RedRover [14]. The latter can be traced back to the control model developed by Kanayama [15].

As with the Argo and RedRover, the implementation of the control was separated into two parts, 1) speed regulation and 2) steering regulation. However, unlike the RedRover, which is a wheeled vehicle and has a natural decoupling of the speed and the turn rate controls, the speed and turn rate are tightly coupled for the tracked drill. Extra care is taken to ensure valid controls, i.e., controls that are within the physical capability of the drill, are issued to lower level controllers. After a waypoint list is received by the waypoint controller, the controller estimates the desired instance speed at each of the waypoint. The estimation is carried based on the heading change rate at the waypoint. For the sake of simplicity, the relative turn rate, ω, for each of the waypoints is approximated as being proportional to the maximum curvature or turning radius, ρ, associated with the trajectory for the region around the waypoint. This is computed using the method descried in [16]. The waypoint speed (v) is hence:

$$v = \rho\omega. \tag{5}$$

Waypoint speed is the basis for the controlled vehicle speed. Since the waypoint speed is estimated beforehand with little consideration of the runtime vehicle behavior, it is difficult to ensure its validity at the runtime. Therefore, the controlled vehicle speed is further verified against the controlled vehicle turning rate. If the controlled vehicle speed exceeds the maximum allowed speed for the specific turning rate, the controlled speed is reduced. The maximum allowable speed for a given turning rate is computed as:

$$v = -(\frac{V_{max}}{\dot{\theta}_{max}})\dot{\theta} + V_{max}. \tag{6}$$

The V_{max} is the maximum allowed velocity and the $\dot{\theta}_{max}$ is the maximum allowed yaw rate of the drill.

Steering regulation is performed with proportional and integral control. The control strategy is designed around error corrections. We define three errors, vehicle heading error, $e(\theta)$, waypoint heading error, $e(\delta)$, and cross-track error, $e(x)$, in the path tracking process. The waypoint controller constantly monitors these three errors and issues steering command to correct them. The resulting control law for turning rate has 6 tuned parameters $(K_\theta \ldots \alpha_x)$ and can be written as:

$$\theta s = \left(K_\theta + \frac{\alpha_\theta}{s}\right) e(\theta) + \left(K_\delta + \frac{\alpha_\delta}{s}\right) e(\delta) + \left(K_x + \frac{\alpha_x}{s}\right) e(x) \tag{7}$$

As shown in Fig. 6, the vehicle heading error θ is defined as the angular difference between the vehicle heading and the vector defined by the vehicle position, the current tracking waypoint and the following waypoint. In our setting, the current tracking waypoint is always ahead of the vehicle in the direction that the vehicle travels. Correcting the vehicle heading error pulls vehicle towards the direction of the path. The desired heading direction is jointly defined by the vehicle position, the current tracking waypoint and the following

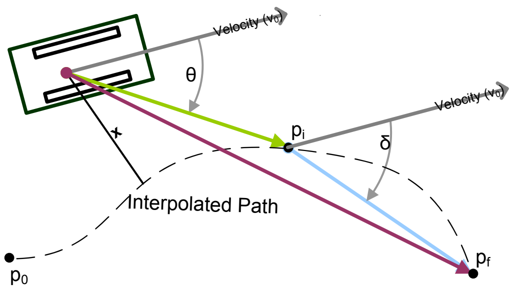


Fig. 6. Vehicle waypoint path following error types. The controller compensates position and orientation between an initial point (p_0), intermediate point (p_i) and final point (p_f) by factoring cross-track error (x) to the current smooth interpolated path (dashed line), and orientation differences between the current and forward waypoints (angles θ and δ).

waypoint. In this setting, the desired heading direction is a weighted average between the vector defined by the vehicle position and the current tracking waypoint and the vector defined by the vehicle position and the following tracking waypoint. It can be described as:

$$\theta = \frac{d}{D}\theta_1 + \frac{D-d}{D}\theta_2. \tag{8}$$

D is the distance between the current tracking point and the previous waypoint. d is the distance between the current vehicle position and the previous waypoint. θ_1 is the angle between the vehicle heading and the vector defined by the following waypoint and the vehicle position. θ_2 is the angle between the vehicle heading and the vector defined by the current tracking waypoint and the vehicle position. This technique provides one step look-ahead for the waypoint tracking. It also removes abrupt changes in the error measurement during the transition of the tracking waypoint.

The waypoint heading error is defined as the angular difference between the vehicle heading and the desired heading at the tracking waypoint. The desired heading at a waypoint is then defined as the vector that connects the waypoint and its successive waypoint.

The cross-track error is defined as the distance between the vehicle position and the planned path. Even though paths are represented as waypoint lists, it is problematic to define the cross-track error as the distance between the vehicle position and the tracking waypoint. As this inevitably introduces sudden changes in the error reading while the vehicle transits from one tracking waypoint to another and brings disturbances to the control. We solve this problem by further interpolating the planned path with high order splines and then take fine discritisation in the interpolation. We then measure the cross-track error as the distance between the vehicle position and the nearest point from the fine-grained path interpolation. This gives accurate reading in the cross-track error and avoids all abrupt changes in error reading due to transition of the tracking waypoint.

V. Experiment Setup and Performance Measure

The tramming method is currently deployed at an iron-ore mine on an autonomous blast-hole drill rig. It has been used in autonomous production operations over a variety of bench types (e.g., Fig. 7). The performance of the control method is shown in path following and drill positioning experiments as measured using the on-board D-GPS position when the machine came to a rest, as compared to desired points as planned and surveyed in advance by mine staff.

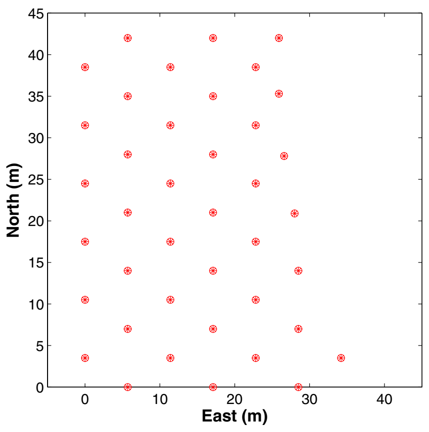


Fig. 7. A sample drill hole pattern (note the regular ordering)

A. Vehicle Configuration

The blast-hole drill rig weighs 98-ton and is 13 m by 5.8 m and has a 17 m tall mast. Navigation is preformed by a Topcon D-GPS suspended on the mast and two generic track encoders. It is suspended on two large tracks and turns by skid-steering. The system has high actuator authority being driven by hydraulics (500 psi) and powered by a 1000 HP diesel motor.

B. Path following

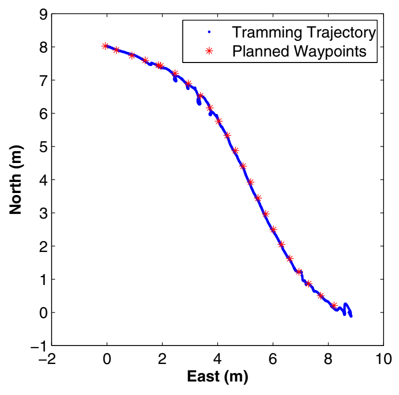
The path planner generates paths that connect drill holes in a pattern. The machine then trams along this path. Motion is constrained by two competing objectives. First, it is desirable to drive as quickly as possible to meet production requirements. However, it is important that the drill does not deviate significantly from the prescribed path given the presence of obstacles and crests. As shown in the Fig. 8 for tramming both a straight and curved path.

C. Drill Mast Positioning

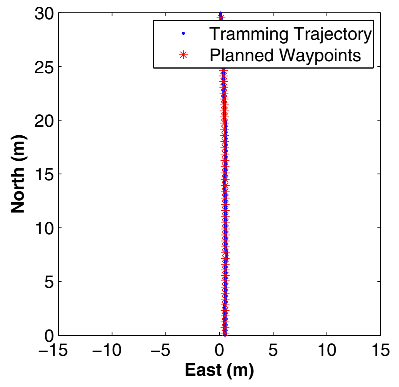
Drill mast positioning is one of the most important criterion of drill operation performance. The feed-forward controller is able to partially compensate for the suspension compliance and the resulting mast sway. Tests with D-GPS positioning have errors <15 cm on average and maximum errors of < 30cm (see Fig. 9). This is particularly notable since this is less than the system's positioning variance (recall the GPS antenna is on top of the swaying mast), thus confirming the utility of a feed-forward approach.

D. Path Planner Performance

The two properties of the path planner we focused our experiments on are: 1) the coverage of all possible trajectories, and 2) the number of node-explorations before a path is



(a) Turning around a bend



(b) Straight line following

Fig. 8. Path following examples: In (a) some turning is involved as the drill moves around a bend in a row-shifting maneuver that is used frequently in production. Note the mast vibration is evident by the circular 'wiggle' present. In (b) a 30 m long straight line trajectory is shown with minimal deviation. These two figures demonstrate the accurate path following performance.

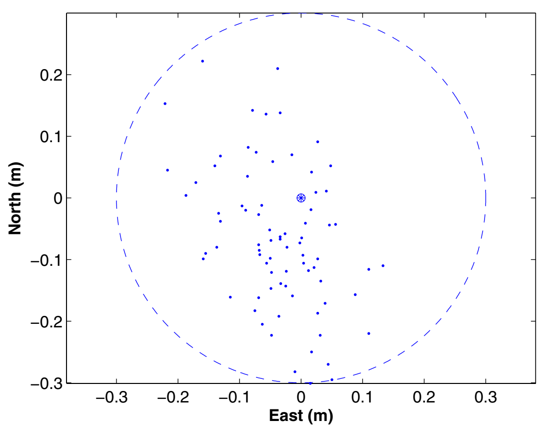
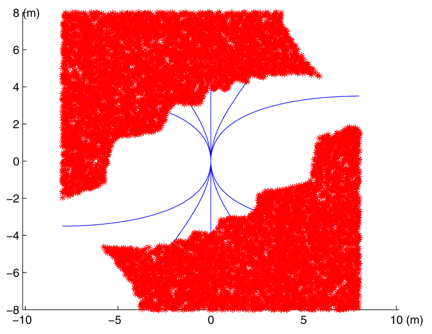


Fig. 9. Drill mast positioning with <15 cm average error.

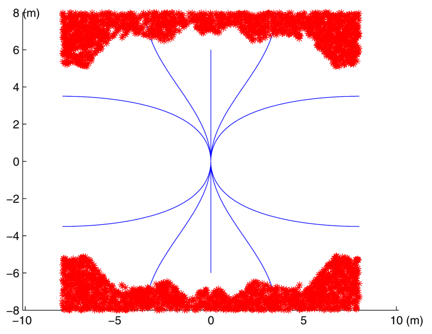
found. The first property represents the completeness of the algorithm, and the second property represents the efficiency of the algorithm.

The Monte Carlo method is used to determine the path coverage. The vehicle starts at (0,0), with heading $\theta_{start} = 0$. Fig. 10 shows the reachable region in the (-8:8, -8:8) m square with the final heading $\theta_{end} = 4 * \pi/5$ and $\theta_{end} = \pi$ after one depth of search node expansion (with 20,000 points are generated in each case). The allowed minimum turning radius for each trajectory is 2 m, the same as the minimum turning radius of the primitive trajectories. Fig. 12 shows the relation between the coverage and the heading difference between the starting position and the final position. In this graph, it can be seen that the unreachable area grows rapidly as the heading difference approaches π. Fig. 11 shows the reachable region in the (-8:8, -8:8) square with the final heading $\theta_{end} = \pi$ after two depth of search node expansion. It can be seen that there is a 100% coverage in this setting.

To test the search efficiency, we expand the end point area to (-20:20, -20:20). 20,000 points are randomly generated in this area as destination points with random heading range



(a) Typical case: Final Heading $\theta_{end} = 4 * \pi/5$



(b) Extreme case: Final heading of $\theta_{end} = \pi$

Fig. 10. Search coverage for the extended state lattice method (compare to Fig. 3). The starting heading is $\theta_{start} = 0$. Shaded areas are reachable in the 8 m by 8 m square (from a starting point of (0,0) after one depth of search node expansion. The later case requires a larger turn and hence shows less coverage.

from $-\pi$ to π. The starting point is (0,0) and the starting heading θ is 0. Fig. 13 shows the histogram of the number of nodes examined before a path is found. Combined with the search coverage results presented above, we conclude our simple breadth-first search is capable of producing desirable results for our application.

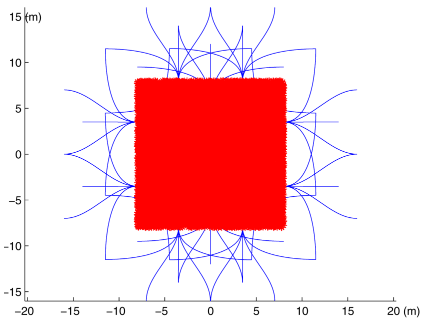


Fig. 11. Search coverage (ref Fig. 10 for a final heading of $\theta_{end} = \pi$ after *two* search node expansions

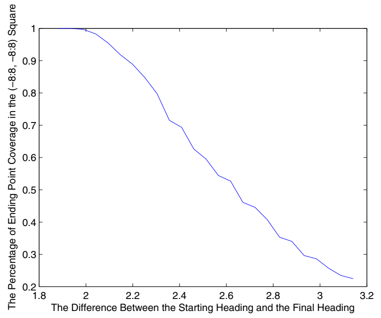


Fig. 12. The search coverage as a functions of total heading variation

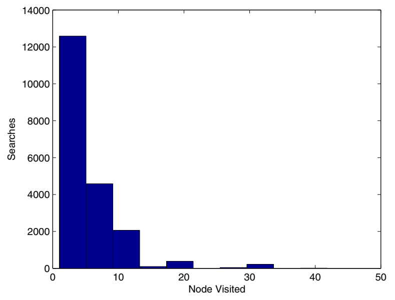


Fig. 13. The histogram of the number of nodes examined for each search. This shows path planning efficiency as most cases did not require extensive searching.

VI. Conclusion and Future Work

A software architecture, a control mechanism and a path planner for an autonomous tracked vehicle are presented. Coverage is important in open environments as there are no roads to follow. The extended state lattice method uses operational data to achieve rapid coverage of the state-space. The tramming controller utilizes a modified waypoint following path tracking technique specialized for high-inertia of the vehicle. The method is implemented as a set of modules and deployed on a 98-ton autonomous blast-hole drill rig in production mining environments.Experimental results include complete autonomous operation in production drill hole patterns. We demonstrated the designed controller is capable of satisfying the two major requirements in mining production in real-time: maintaining good overall path tracking performance and realizing accurate drill hole positioning. Field tests indicate that both the average cross-track errors and average positioning errors are <0.15 meters.

Future developments include further system refinement to improve operating efficiency.

VII. Acknowledgements

This work is supported by the Rio Tinto Centre for Mine Automation and the ARC Centre of Excellence program funded by the Australian Research Council (ARC) and the New South Wales State Government. The authors also acknowledge the support of Charles McHugh of Rio Tinto.

References

[1] H. L. Hartman, *SME Mining Engineering Handbook*. SME, 1992.

[2] P. Elinas, “Multigoal planning for an autonomous blasthole drill,” in *19th International Conference on Automated Planning and Scheduling*, Sept. 2009, pp. 342–345.

[3] T. Howard and A. Kelly, “Optimal rough terrain trajectory generation for wheeled mobile robots,” *International Journal of Robotics Research*, vol. 26, no. 2, pp. 141–166, Feb 2007.

[4] J. L. Martínez, A. Mandow, J. Morales, S. Pedraza, and A. García-Cerezo, “Approximating kinematics for tracked mobile robots,” *Int. J. Rob. Res.*, vol. 24, no. 10, pp. 867–878, 2005.

[5] A. T. Le, D. Rye, and H. Durrant-Whyte, “Estimation of track-soil interactions for autonomous tracked vehicles,” *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, vol. 2, pp. 1388–1393 vol.2, Apr 1997.

[6] Z. Fan, J. Borenstein, D. Wehe, and Y. Koren, “Experimental evaluation of an encoder trailer for dead-reckoning in tracked mobile robots,” in *Proceeding of the IEEE International Symposium on Intelligent Control*, Monterey, CA, 1995, pp. 571–576.

[7] Z. Shiller, W. Serate, and M. Hua, “Trajectory planning of tracked vehicles,” *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, pp. 796–801 vol.3, May 1993.

[8] M. Ahmadi, V. Polotski, and R. Hurteau, “Path tracking control of tracked vehicles,” in *Proceedings of the International Conference on Robotics and Automation (ICRA 2000)*, vol. 3, 2000, pp. 2938–2943 vol.3.

[9] G. Wang, S. H. Wang, and C. W. Chen, “Design of turning control for a tracked vehicle,” *IEEE Control Systems Magazine*, vol. 10, no. 3, 1990.

[10] R. A. Knepper and A. Kelly, “High performance state lattice planning using heuristic look-up tables,” in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, October 2006, pp. 3375 – 3380.

[11] P. Giordano, M. Vendittelli, J.-P. Laumond, and P. Souéres, “Nonholonomic distance to polygonal obstacles for a car-like robot of polygonal shape,” *IEEE Transactions on Robotics*, vol. 22, no. 5, pp. 1040–1047, oct 2006.

[12] T. Fraichard and J.-M. Ahuactzin, “Smooth path planning for cars,” in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, vol. 4, 2001, pp. 3722–3727.

[13] Q. P. Ha, T. H. Tran, S. Scheding, G. Dissanayake, and H. F. Durrant-Whyte, “Control issues of an autonomous vehicle,” in *22nd International Symposium on Automation and Robotics in Construction ISARC*, Ferrara, Italy, September 2005.

[14] T. Henderson, M. Minor, S. Drake, A. Hetrick, J. Quist, J. Roberts, H. Sani, R. Bandaru, M. Rasmussen, A. Collins, Y. Sun, Suraj, X. Fan, E. S. Louis, S. Mikuriya, and K. Dean, “Robust Autonomous Vehicles,” July 2007, dARPA Urban Challenge Technical Paper.

[15] Y. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi, “A stable tracking control method for an autonomous robot,” in *Proceedings IEEE International Conference on Robotics and Automation*, vol. 1, 1990, pp. 384–389.

[16] R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision (Volume II)*. Prentice Hall, June 2002.