Unmanned Systems, Vol. 5, No. 1 (2017) 45–57

DOI: 10.1142/S2301385017500042

UNMANNED SYSTEMS
http://www.worldscientific.com/worldscinet/us

Experimental Test of Unmanned Ground Vehicle Delivering Goods Using RRT Path Planning Algorithm

Yiqun Dong*,†,‡, Youmin Zhang†,§, Jianliang Ai*

*Department of Aeronautics and Astronautics
Fudan University, Shanghai, China

†Department of Mechanical and Industrial Engineering
Concordia University, Montreal, Canada

This paper presents the experimental test of an unmanned ground vehicle delivering goods. Configuration and motion equations of the vehicle are illustrated, drivers for the vehicle motion control are introduced. In the presence of obstacles, the collision-free path connecting the vehicle from the start to the goal position is planned using Rapidly-exploring Random Tree (RRT) algorithm; collision detection, nodes selection, tree expansion, and path generation of the RRT are presented, the path optimization approach is discussed. To grip the goods, vehicle mechanical arms are manipulated based on the inversed kinematics, some control flow of the arms deployment for interacting with the vehicle motion control is applied. Experimental test of the vehicle delivering goods in face of static obstacles is presented; test result validates the applicability of the proposed framework.

Keywords: Unmanned ground vehicle; goods delivery; RRT path planning; mechanical arms.

US

1. Introduction

Recent years have witnessed an increasing interest in applying robots to modern industries and human daily lives. Mechanical arms have long been used in factory production lines; unmanned aerial vehicles serve in military missions for battleground reconnaissance; autonomous intelligent ground/aerial vehicles explore and send back vital information for search and rescue in disastrous circumstances (earthquake, fire, etc.); and even for our houses, autonomous machines are appearing on the market (self-walk floor scrubbers).

A team of researchers with the Networked Autonomous Vehicle (NAV) laboratory at Department of Mechanical and Industrial Engineering in Concordia University is also working on the design and implementation of unmanned ground vehicles (UGVs) and unmanned aerial vehicles (UAVs). Subjects of developing these vehicles for autonomous driving, automatic parking, and cooperative formation control [1, 2] have been investigated. Specifically, in response to the advocate of Amazon couriering goods via quadrotors, the topic of using UGV for goods delivery has been initiated. Generally, in a constrained environment, the vehicle is expected to go to the goods, load it, transfer it to the target position, and unload it. Given certain obstacles in the environment, certainly, the vehicle is also expected to circumvent all the obstacles.

There are three technical challenges in the UGV goods-delivery problem. First, a path planning algorithm needs to be used. Given mathematical representation of the working space, this algorithm is expected to find a collision-free path which can guide the vehicle to go to the goods, and to move along with the goods to the target region. Second, control of the vehicle gripping devices (mechanical clamps/arms) should be investigated; on the one hand it should reach out to grip and hold the goods, and on the other hand it should interact with the motion control of the vehicle (the vehicle should go to the goods first, and then deploy the gripper, for

Received 13 June 2016; Revised 9 January 2017; Accepted 9 January 2017; Published 17 March 2017. This paper was recommended for publication in its revised form by editorial board member, Yunfeng Zhang.
Email Addresses: ‡*yiqundong10@fudan.edu.cn*, §*ymzhang@encs.concordia.ca*

instance). Lastly, an online motion controller is called. Given the desired path exported from the path planning algorithm, trajectory tracking of the UGV is completed using this controller.

In literature, the first highlight of the robot path planning has been the topic of many research efforts. Cell decomposition method is proposed in [3]; by partitioning the environment into a finite number of regions (cells), path planning is converted to the problem of finding adjacent cells which contain the vehicle start and goal positions. In [4–6] a roadmap approach is introduced; a network of collision-free path is gradually expanded, and a concatenate of the paths on this roadmap which connects the vehicle start and goal positions is defined as the final path. One mathematically-elegant approach proposed for the path planning is the artificial potential field (APF) method [7–10]. A potential function field is generated based on the working space, wherein obstacles represent repulsive and goal induces attractive forces; the vehicle moves along the direction which being locally defined as the maximum gradient drop of the potential, and slides towards the goal. Lastly an approach named as A* (and its variants) is widely used [11–14]; this algorithm discretizes the environment first, based on the pre-defined heuristic function it determines the order in which surrounding grids are visited; as the visited area expands, the algorithm will quit when the goal position is contained in the inspected area.

One intractable problem of the above mentioned methods, however, is the relatively high computational load. Even for the APF method wherein the real-time computation is comparably less-intensive (as it demands a search for the maximum gradient-drop only), generating the potential field remains a very challenging task. Also for most of the algorithms, especially APF and A*, some manipulative factors are involved. In APF the potential field is generated using a function/coefficient biasing the obstacles and goal, and in A* the heuristic function needs to be predefined. While these definitions might be suitable for some cases, the effect of them to other scenarios is unpredictable. Lastly and most importantly, generally only kinematics model of the vehicle is considered in above methods. It is either not applicable (cell decomposition) or complicated (roadmap, APF, A*) to consider the nonholonomic constraints of the vehicle (actuator dynamics, minimum turning radius, etc.). In some cases the generated path is too odd (sharp turning, lateral movement, etc.) for the vehicle to execute.

A more applicable choice to address the path planning problem is the Rapidly-exploring Random Tree (RRT) algorithm. First proposed in [15], RRT has attracted a significant amount of research attention; to name only a small part of which readers may refer to [16–19]. The RRT imitates the growth of a natural tree, and at each step of the tree expansion, only collision-free and dynamically-feasible (subject to vehicle nonholonomic constraints) paths/nodes are added onto the tree. The tree-growth stops when a certain node on the tree reaches the goal, and the final path is generated by tracing back from the goal to the start (root of the tree). Computational complexity of the RRT is relatively low, and it has the inherent capability of taking into account the nonholonomic constraints of vehicle dynamics/kinematics. In our work the RRT algorithm is used for the vehicle path planning.

As for the latter two challenges, mechanical arms of the modeled vehicle are actuated by electric control servos, and the arms control involves some inversed kinematics. In goods-load stage, the arms are expected to reach out to the goods, grip, and lift it up. In final stage of the goods-unload stage, this process is reversed; the arms will lower and unleash the goods. Input for each of the arms servos is decided based on the inversed kinematics. For vehicle motion control, two different drivers are used; one is for the RRT tree-growth, and the other for online vehicle trajectory tracking. Details of these two challenges will be presented later in following parts of this paper.

Structure of this paper is organized as follows. In Sec. 2, configuration and motion equations of the vehicle are introduced, inversed kinematics for the mechanical arms control are included. In Sec. 3, two drivers for the vehicle motion control are presented; advantages and disadvantages of each are discussed. Overall framework of the path planning algorithm is illustrated in Secs. 4, and 5 presents an experimental test of UGV delivering goods. Conclusions of this paper can be found in Sec. 6; several issues that warrant authors' future attention are also listed in this section.

2. Vehicle Configuration and Motion Equations

The vehicle used herein is the so-called Quanser Ground Vehicle, see Fig. 1. Length and width of the vehicle are 66 cm and 40 cm, respectively. The vehicle is driven by two front wheels, each actuated by an electric encoder; rear wheel is used for balance support only. Six joints (control servos) are used for the manipulation of the mechanical arms. First joint (J_1) at the bottom is used to bias heading of the arms base, joint to the top (J_5) is for the gripper, and the combined effect of other joints (J_2, J_3, and J_4) elevates/extends the arms. In this section we mainly discuss the mathematical model of vehicle motion equations and inversed kinematics for the arms control.

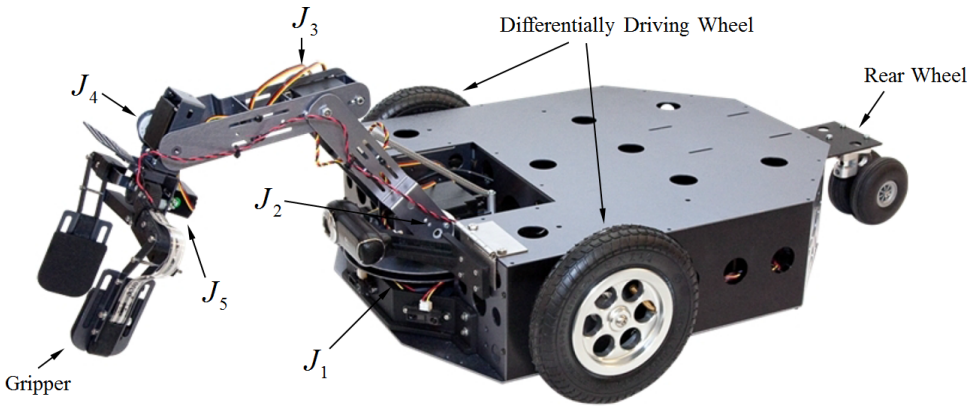


Fig. 1. UGV configuration.

2.1. *Vehicle motion equations*

See Fig. 2; given position (x, y), heading angle (θ), and speed of the vehicle right/left wheels (V_R, V_L), motion equations of the vehicle are:

$$\begin{cases} V_C = (V_R + V_L)/2 \\ \dot{x} = V_C \cos\theta \\ \dot{y} = V_C \sin\theta \\ \dot{\theta} = (V_R - V_L)/W. \end{cases} \quad (1)$$

Note that in (1) $W = 40$ cm is the wheel basis, which is defined as the distance between the two driving wheels. Also, speed of the vehicle V_C is defined by averaging the right and left wheel speeds; given that the vehicle is a three-dimensional rigid body, this definition puts the reference point (pivot) of the vehicle motion to the center of the axle connecting the two driving wheels.

Based on (1), motion of the vehicle can be controlled by manipulating the speed of two driving wheels (V_R, V_L). Speeds of these two wheels are controlled by electric encoders; prior to the motion control, a mapping function from the encoder input to the actual wheel speed is decided, see Fig. 3. Note that encoder input of the wheel is restricted between 0.05–0.10; for input below 0.075 (0.05–0.075) the wheel reverses, while for input above 0.075 (0.075–0.10) the wheel moves forward. At an incremental step of 0.0025, encoder input (for both wheels) is gradually increased, and the speed of the vehicle is calculated (black circles). A dead zone of the encoder input exists between 0.0725–0.0775 (red line), excluding which, the functional relation between the wheel encoder input and vehicle speed is regressed (green line for moving forward and blue line backward).

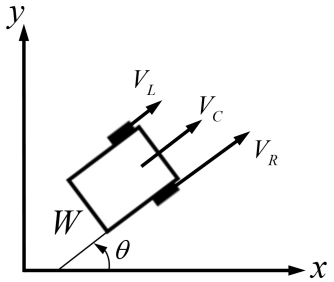


Fig. 2. UGV kinematics.

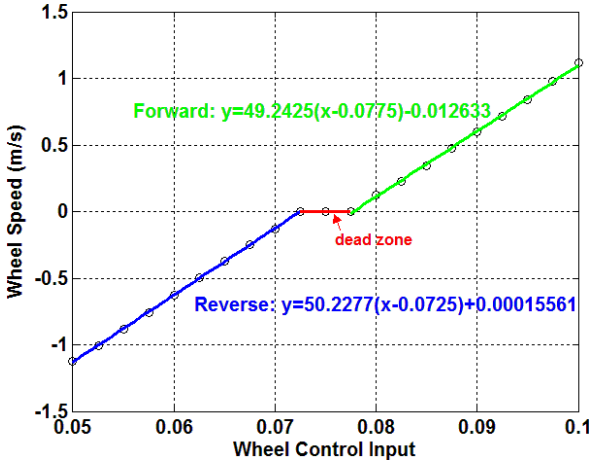


Fig. 3. Mapping relation from UGV input to wheel speed.

Given Eq. (1) and Fig. 3, theoretically the UGV can turn without moving its position; this can be achieved by commanding one driving wheel forward and the other wheel backward. In our test to simulate the minimum turning radius constraint, no reverse of the driving wheels is allowed; i.e., the minimum turning radius of the vehicle is half the distance between the two driving wheels. Also for security issues, speed of the vehicle is limited at 20 cm/s.

2.2. *Inversed kinematics for the mechanical arms control*

As shown in Fig. 1, mechanical arms fixed on the vehicle are manipulated by five joints (control servos). The servo at the bottom biases the pointing direction. For our test discussed herein, the arms are aligned with the heading direction of the vehicle, control of the first joint (J_1) is therefore not involved. Also, the servo to the top (J_5) is used for the gripper only, it does not have independent effects on the geometric control of the entire arms system.

The overall plot for the arms kinematics is shown in Fig. 4; on the plot positive Y-axis is aligned with the heading direction of the vehicle, and positive Z-axis points vertically upward. Due to the Quanser design, deflection of joint J_2 is always positive (counter-clockwise on Fig. 4), and joint J_3 negative (clock-wise on Fig. 4). In our test, command signal delivered to the arms controller is the desired position/pitch of the gripper (J_5), based on Fig. 4 we can have:

$$\begin{cases} y_4 = y_5 - l_3 \cos\delta \\ z_4 = z_5 - l_3 \sin\delta. \end{cases} \tag{2}$$

In (2) y_5, z_5 and δ is the position/pitch command; l_3 is the length of the third arm, which in our test is 15.4 cm; y_4 and z_4 is the desired coordinates of joint J_4.

Given the length of the first and second arm ($l_1 = 15.4\,\text{cm}$, $l_2 = 12.3\,\text{cm}$), for the triangular enveloped by J_2, J_3 and J_4, we can have

$$l_1^2 + l_2^2 - 2l_1 l_2 \cos(\pi - \delta_2) = y_4^2 + z_4^2 \tag{3a}$$

and due to Quanser design

$$\delta_2 < 0 \tag{3b}$$

desired deflection angle δ_2 of joint J_3 can be decided.

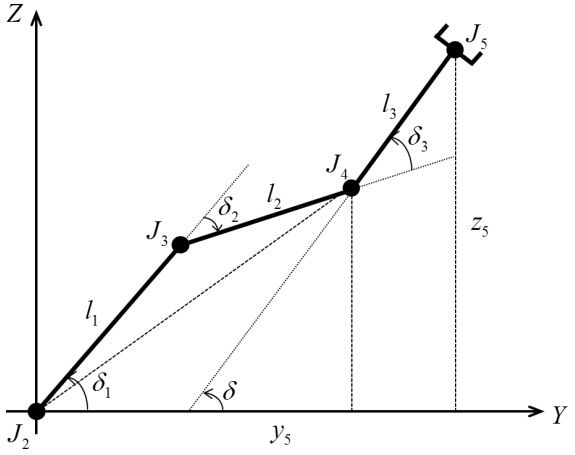


Fig. 4. Inversed kinematics of the UGV mechanical arms.

Given the deflection angle of the first arm (δ_1), the projection of the first and second arms (l_1 and l_2) to Y and Z-axis yields:

$$\begin{cases} l_1 \cos\delta_1 + l_2 \cos(\delta_1 + \delta_2) = y_4, \\ l_1 \sin\delta_1 + l_2 \sin(\delta_1 + \delta_2) = z_4, \end{cases} \tag{4a}$$

again due to Quanser design:

$$\delta_1 > 0 \tag{4b}$$

the deflection of the first arm (δ_1) can be calculated.

Note that pitch of the last arm (J_4) is δ, given δ_1 and δ_2:

$$\delta_1 + \delta_2 + \delta_3 = \delta \tag{5a}$$

we can have

$$\delta_3 = \delta - \delta_1 - \delta_2 \tag{5b}$$

Based on Eqs. (2)–(5), given the desired position (y_5, z_5) and pitch command (δ) of the gripper, deflection angle for each of the electric motor can be decided $(\delta_1, \delta_2, \delta_3)$, and the arms control is completed.

3. Vehicle Motion Controller

Vehicle motion controller is used for two purposes. In the final test it will be called by the vehicle to track planned path, and in the RRT tree-growth stage it will be used for connecting new nodes onto the RRT tree. For online tracking, the pure-pursuit controller is used, and for tree-growth a so-called max-capability turner (MCT) is developed.

3.1. *Pure-pursuit controller*

First proposed and applied in [20], pure-pursuit controller has been widely used for ground and aerial vehicles path tracking, see [21–22]. It is adopted here due to its simplicity, especially as it is an intuitive control laws which has clear geometric meanings. At each moment, the vehicle is modeled as moving along a curved arc which travels to the goal. A conceptual plot of the pure-pursuit controller is shown in Fig. 5. On the plot, the vehicle starts the turning at point O, P is the origin of the vehicle turning arc, r is the vehicle turning radius, and the goal of the vehicle is located at G. Given Fig. 5.

$$\begin{cases} r^2 = d^2 + y^2 \\ r = d + |x| = d - x \end{cases} \tag{6}$$

we can have

$$r = -\frac{x^2 + y^2}{2x} \triangleq -\frac{L^2}{2x}, \tag{7}$$

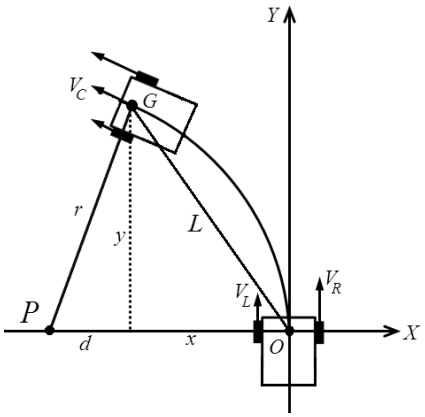


Fig. 5. Pure-pursuit controller geometrics.

wherein $L \triangleq \sqrt{x^2+y^2}$ is defined as the look-forward distance of pure-pursuit controller.

Given the motion equations in (1)

$$\begin{cases} \dot{\theta} = (V_R - V_L)/W \\ V_C = (V_R + V_L)/2 \end{cases} \tag{8.1}$$

and the speed of the vehicle

$$V_C = r\dot{\theta}, \tag{8.2}$$

we can have

$$\begin{cases} V_R - V_L = W\dfrac{V_C}{r} \\ V_R + V_L = 2V_C \end{cases}. \tag{9}$$

Substitute r from (7) into (9)

$$\begin{cases} V_R = V_C\left(1 - \dfrac{W}{L^2}x\right) \\ V_L = V_C\left(1 + \dfrac{W}{L^2}x\right) \end{cases}, \tag{10}$$

which is the desired speed for the vehicle right and left wheels. Given the mapping relation presented in previous section, control input for the vehicle driving wheel encoders can be decided.

In (10) note that x is the lateral displacement of the goal in vehicle body axis, speed of the vehicle is donated as V_C. As for the specifying of the look-forward distance, however, to authors' best of knowledge there does not exist a universal standard. In Ref. [23] it was mentioned that the controller is Hurwitz stable if L is larger than a certain level, which being closely related with the traveling speed of the vehicle (V_C). In some of the research efforts the look-forward distance was determined using experimental tests [21]; for the work discussed herein, wherein vehicle speed is 20 cm/s, via extensive simulation and experimental validations, a look-forward distance of 15 cm is used.

3.2. *Maximum-capability turner*

Another motion controller used here is what the authors refer to as the maximum-capability turner (MCT). A rigid turning mechanism which is similar to the Bang-bang laws from optimal control is used. At each moment, the vehicle is turned at its maximum turning capability (minimum turning radius), until the heading of the vehicle is pointed directly to the goal. Different from the pure-pursuit controller, of which the turning radius is gradually increased at each step of the (as x gradually drops), the vehicle controlled by MCT is rigidly rotated along the minimum turning radius arc.

In the final test stage trajectory tracking of the vehicle is accomplished using the pure-pursuit controller. First, theoretically (and also based on authors' experimental tests) pure-pursuit controller is more stable than MCT, especially as in real test the uncertainties/measurement noise will arise, and in extreme cases an unstable oscillation might be induced by the MCT rigid turning. Apart from the stability issue, pure-pursuit controller is more accurate than MCT. In authors' test the tracking error of pure-pursuit controller is below 3 cm while for MCT it may be high as 10 cm, which again could be due to the rigid turning in MCT.

The MCT controller, however, is suitable for the RRT tree-growth. In this stage, the MCT is used to build up a collision-free path connection from a certain node on the already-existed tree to the random seed. A comparison of the vehicle response history from a certain start to the same goal using both pure-pursuit controller and MCT is shown in Fig. 6. On the plot, *O* is the start point, goal point is *G*; trajectory *I* is navigated using pure-pursuit controller, and *II* is generated from MCT. For trajectory *I*, again the vehicle is modeled as moving along an arc constantly, the pure-pursuit controller generally slides the vehicle along this arc,

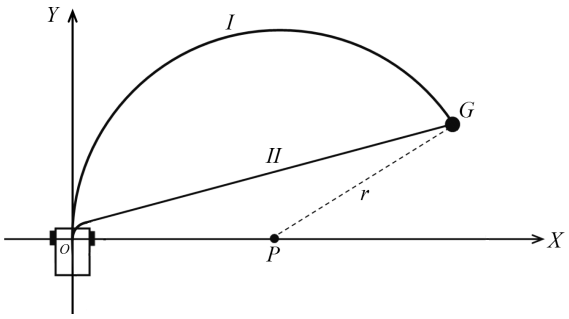


Fig. 6. Comparison between pure-pursuit controller and maximum-capability turner (MCT).

and the vehicle will get to the goal point. For trajectory *II*, however, the MCT first turns the vehicle at its maximum turning capacity towards the goal point, then the vehicle goes straight to the goal point. As shown on Fig. 6, length and covered area of path *II* is smaller than *I*, which by instinct, indicates that there is a lower chance that the vehicle will collide with obstacles. Although this is a very general conclusion, based on trial tests and extensive simulations the RRT tree-growth with MCT is faster than using pure-pursuit controller. While further investigations will be conducted, in our test we use the MCT turner for the RRT tree-growth.

4. RRT-Based Path Planning

Given start/goal positions of the vehicle and certain constraints of the working space (boundaries, obstacles), the path planning algorithm is called to find a feasible path which guides the vehicle from the start to the goal position. We use RRT algorithm in our work. In this section the basic framework of RRT is illustrated; some optimization approach to shorten the overall distance of the planned path is also presented.

4.1. *General RRT algorithm framework*

The RRT method is a sampling-based expansion algorithm. At each step of the tree-growth, given the generated sample (random seed), the growth of the tree is attracted to this seed until a certain branch of the tree can reach the goal. Based on different sampling approaches, roughly two categories of the RRT have appeared: for the original RRT, the tree grows by sampling the system input or configuration; for latter case the input needs to be reversely calculated [15, 16]. In [21, 23] the closed-loop RRT (CL-RRT) is proposed. This method extends the tree by sampling "goal position", and the vehicle is navigated to this goal using a bottom-level controller (MCT, for instance). Practically, computation intensiveness of CL-RRT might be higher than the original RRT (as an iteration loop is included to navigate the vehicle toward the goal), however nonholonomic constraints are taken into account in the navigation. In this paper the RRT algorithm follows the CL-RRT structure.

Pseudo-code of the RRT algorithm used herein is illustrated in Table 1. The tree-growth is initiated at the root of the start (line 0); the algorithm detects whether or not current tree branches (nodes) can reach the goal (line 1). If not, a random seed within the working space is generated

Table 1. Pseudo structure of the RRT algorithm.

Algorithm Tree_Grow ()

0: Add start point to root of the tree.
Repeat
1: If the newly added node can tree can "see" the goal point, connect this node and goal and break.
2: Generate a random seed within the work space.
3: **for** all the existed tree nodes:
4: Calculate the Eucliean distance between node and seed.
5: Add Eucliean distance to the distance stored in node.
6: Find the parent node with minimum total distance.
7: **end for**
8: **Navigate** the vehicle from parent node to seed until:
9: The vehicle is out of the working space; or
10: Vehicle collides with obstacles; or
11: Maximum length of the vehicle movement is met; or
12: Vehicle falls into certain neighboring region of seed.
13: **end Navigate**
14: Add the vehicle stopped point to the tree.
15: Store:
Node distance between the new node and root;
Traveling distance between new node and root;
Parent node number.
into the new node.
end Repeat
16: Trace back from finally-added node to the tree root.

(line 2), nodes already-existed on the tree are inspected, and the parent node to be expanded is selected (line 3–7). Bottom level controller is called in line 8–13, which is used to navigate the vehicle from the node selected in line 3–7 to the seed generated in line 2, until certain stop conditions are met (line 9–12). The vehicle stop point is added onto the tree as new node (line 14). Certainly, some information pertaining to node distance, node numbers, etc., is stored in each of the newly added node (line 15). After the entire tree expansion is completed, in line 16 the final path is generated by tracing back from the goal to the tree root (initial vehicle start position). In following contents we detail the algorithm.

4.2. *Vehicle navigation and collision detection*

Vehicle navigation is included for two purposes. At the beginning of the tree-growth, for each of the newly added node (including the initial tree root), motion controller (MCT) is called to drive the vehicle from this node to the goal (line 1); once a collision-free path can be connected, the vehicle is defined as being feasibly able to "see" the goal, and the tree-growth is completed. Different from most of the RRT applications of which the tree-grow will not stop until a certain branch (node) falls into the neighborhood of the goal, this "see-to-stop" tree expansion strategy is proved to be faster.

Another usage of the navigation is in the tree expansion stage (line 8–13). Given the random seed generated in line 2, and the to-be-expanded parent node selected from line 3–7, the MCT controller is adopted to simulate the vehicle response from the parent to the seed, until certain stop conditions are met (line 9–12). MCT controller has been discussed in previous sections; here we mainly discuss the stop conditions.

For line 12, the outcome is obvious: since the vehicle is close enough to the seed, the navigation can be stopped. In line 11, the maximum step for the tree expansion represents the "resolution" of tree exploration. A smaller maximum-length corresponds to a more detailed inspection of the working area, although more iteration might need to be included to complete the tree-growth, and the computational load can be higher. In authors' test, a maximum length of 15 cm is considered to be a balance between tree exploring and computational demanding.

Lines 9–10 belong to the collision detection work. For authors' test, both the UGV and obstacles are geometrically convex, of which the position is restricted within a concave working space. See the conceptual plot in Fig. 7. In the figure, working space boundaries are represented by green lines, and the yellow cubic denotes obstacles. Totally three collision cases (stop conditions) might happen. In case *I*, part of the UGV body goes out of the working area; given the four coordinates of the convex UGV body, and the four coordinates of the concave working area, this case can be checked.

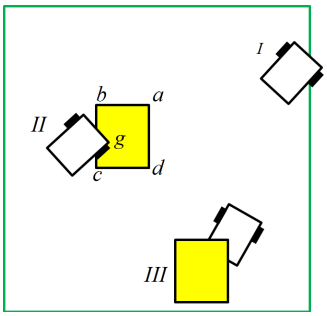


Fig. 7. Conceptual plot for collision detection.

Cases *II* and *III* are classic geometric rectangular intersection problem. In case *II*, given the four coordinates of the obstacle (a, b, c, d) and the inspected point laid on UGV body (g), by checking:

$$\begin{cases} \Pi_1 = \boldsymbol{ab} \times \boldsymbol{ag} \\ \Pi_2 = \boldsymbol{bc} \times \boldsymbol{bg} \\ \Pi_3 = \boldsymbol{cd} \times \boldsymbol{cg} \\ \Pi_4 = \boldsymbol{da} \times \boldsymbol{dg} \end{cases}, \tag{11}$$

wherein vector cross-multiplication is used ($\boldsymbol{xy}$ indicates the vector pointing from x to y,); for all the four cross-multiplied values in (11), if the algebraic signs are identical (positive for a counter-clockwise *abcd* sequence on the figure), the inspected point (g) is laid within the obstacle. Same criteria can be applied to case *III*, which indicates that the obstacle is laid inside UGV. For both cases a collision is detected, and the navigation simulation will stop.

4.3. *Seed generation, node selection, and path connection*

In line 2 of Table 1, a random seed within the working space is generated. For RRT, this seed represents the (random) direction to which the tree will expand. In literature several research efforts focused on manipulating the seed generation to improve the RRT performance [24–26]. One "overfit" problem, however, might arise; although they might have the claimed effect for a specific working space, compatibility of these seed generation to other scenarios is doubtful. In this paper, the author follows the original pure-random manner in the seeds generation.

In line 15 of the algorithm, some information is stored in each of the newly added node, which includes parent node number and the distance between the added node and the tree root. This information is used on one hand for the

concatenate of the final planned path, and on the other hand for the selection of the to-be-expanded parent node, as shown in lines 3–7. In literature some researches have been conducted to bias the selection of the parent node [27, 28]. In this paper the authors adopt the approach similar to [15]; given the distance stored in each node, and the Eucliean distance between the parent node and the seed, total distance from the seed to the tree root is obtained, and the algorithm will select the node via which the seed connects to the root at lowest distance.

After the repeat loop in Tree-Grow algorithm is completed in Table 1, a certain branch can feasibly "see" the goal. The final path is connected using the information stored in line 15. Note that for each parent node in the tree, it can have multiple children nodes (children nodes connect to the tree root via the parent node), one child node, however, maps to only one parent node. Given the lastly added node, by tracing the parent node to the start root of the tree, the final feasible path is connected.

4.4. *Path optimization and waypoints generation*

Based on the Tree-Grow algorithm in Table 1, a feasible (collision-free, subject to nonholonomic constraints) path can be found. Some optimization efforts are included. On one hand this optimization is expected to shorten the overall distance from the start to the goal position, and on the other hand it helps to smooth the path curvature, which improves the tracking accuracy in the final test.

In literature several approaches have been proposed for the optimization in RRT, among which specifically in [28], an improved version named as RRT* is mentioned. The RRT* mainly takes effect in the tree-growth process. For each step, apart from selecting the parent node to expand, it also tries to connect other tree nodes (within a certain neighborhood of newly added node) via the new node to the tree root; if for a certain node the newly connected distance is smaller than the original one stored in the node, parent of this node will shift to the newly added node, the tree structure is rewired. Based on extensive simulations, however, on one hand computation intensiveness of RRT* is higher (it will get worse with larger node number), and most importantly this algorithm takes effect primarily when the tree nodes number is high enough (since the neighborhood mentioned above is defined based on the number of the tree nodes); in the test discussed herein this approach will not be adopted.

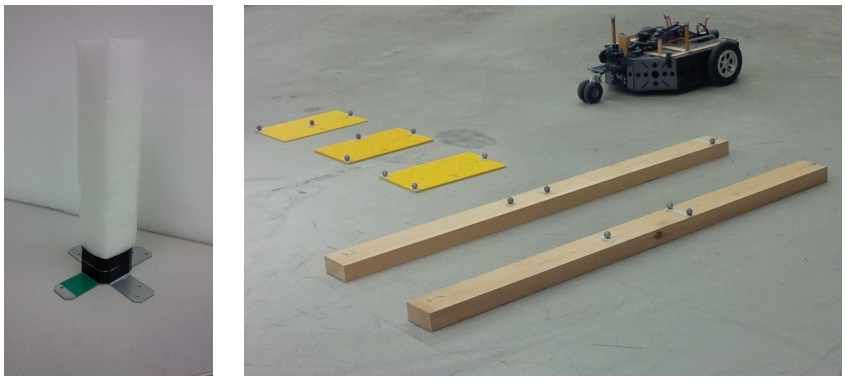
What is used is similar to the approach in [7, 29]. Since the feasible path has been found (which consists of multiple tree nodes connected by vehicle response history from MCT simulation), the path is optimized by randomly picking up two points on the path and trying to connect them using the MCT directly. If there is a collision-free connection between these two points, and if the distance of this new connection is smaller than the original one, the original path will be replaced by this new connection. If this collision-free connection does not exist or the new distance is larger, an empty trial is returned. Given that we are always trying to tune the original path using new lower-distance connections, overall distance of the path evolves at a non-increasing manner, and this process is repeated until maximum empty-trial number is met. In our test, this number is 25.

After the above optimization process, one foreseeable result is that the nodes (which may include original tree nodes and the randomly picked-up points along the path) are scattered along the path (connected by the trajectories from the MCT simulations), and the distance between two consecutive nodes can be different. Waypoints are added along the planned path. Starting from the vehicle start position, each point along the path is examined. If the distance between this point and the previous waypoint (vehicle start is defined as the first waypoint) is larger than a predefined level, this point is added as a new waypoint, and this process will repeat till the end of the feasible path (goal). In this way, the final path is divided by multiple segments, each enveloped by two consecutive waypoints, between which the distances are equal. Based on authors' test, this can improve the tracking accuracy in the final experiment. In authors' test the distance between two consecutive waypoints is the same as vehicle look-forward distance (15 cm).

5. Experimental Test

Final experimental test is conducted in the NAV lab. One important component of the lab is the OptiTrack indoor positioning system. Infrared cameras are used to capture reflective markers fixed on the vehicle. Given location of the marker within camera's imaging plane and the position/orientation of each camera, coordinates of the marker can be decided; with multiple markers fixed on the vehicle, position/heading of the vehicle can be calculated. Totally 24 cameras are used in the NAV lab, which can roughly track the position within a 4 m $\times$ 4 m square.

The goods used in the test is handmade by the authors, see Fig. 8(a). Main body of the goods is made of air foam, and the four stands fixed at the bottom are steel. Height of the goods is 30 cm, base width is 10 cm. Given the low force of the gripper, weight of the goods is restricted below 200 g. Obstacles used in the test are shown in Fig. 8(b); length and width of the long obstacles are 110 cm and 12 cm, while for short yellow ones it's 30 cm and 20 cm, respectively. Note



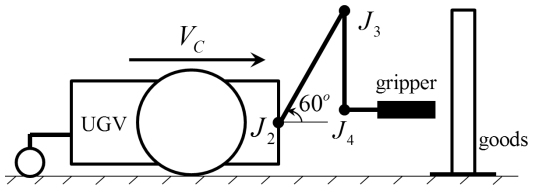
(a) (b)

Fig. 8. (a) Goods to be delivered in the test. (b) UGV and the obstacles.

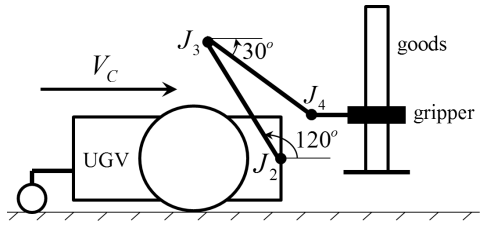
that markers are fixed on the obstacles and the goods, positions of which can then be measured in OptiTrack.

Initial settings of the test are illustrated in Fig. 9. To ensure tracking of the OptiTrack, working space of the test is a 3 m × 3 m square. Initial position of the vehicle is located at top right, and goal position for the goods-delivery is to the bottom left. In the test, three obstacles exist in the working space (black rectangular), one is long (ID 2), and the other two are short yellow cubic (ID 3, 4). Also on the plot, the object to be delivered is represented by the red dot (edged by thick black line).

Note that in the beginning of the test, we initialize the state of the mechanical arms first. See the "go-to-grip" configuration in Fig. 10(a). We keep the pitch of the gripper horizontal ($\delta = 0$), deflection of the first arm $\delta_1 = 60°$, and the second arm is put to vertical ($\delta_2 = -150°$). The goods load range can be calculated based on this configuration. As in Fig. 10(a), from J_4 to the vehicle is around 8 cm



(a)



(b)

Fig. 10. Two configurations of the mechanical arms: (a) "go-to-grip" and (b) "grip-to-go".

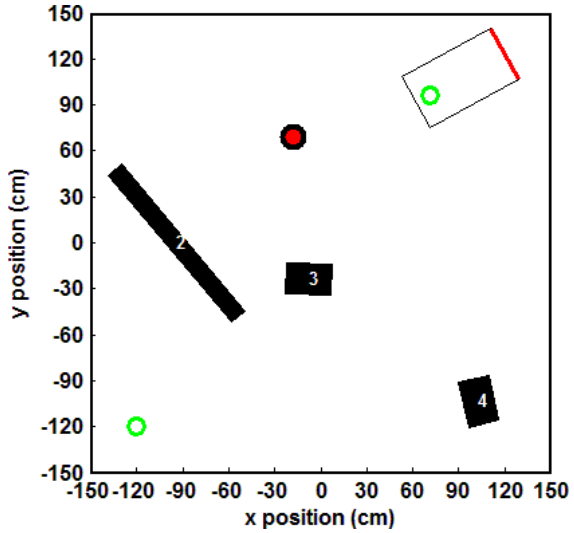


Fig. 9. Initial settings for the experiment test.

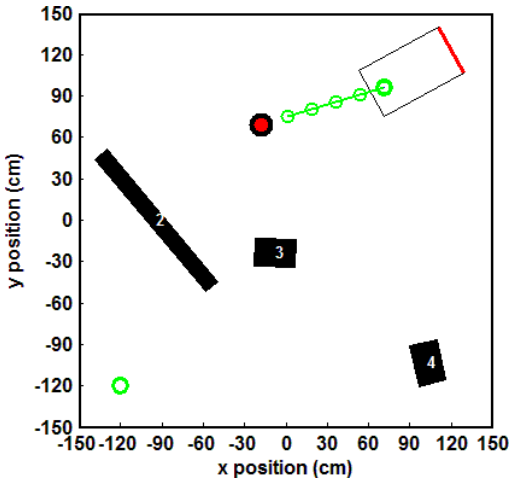


Fig. 11. Planned path for UGV moving to pick up the goods.

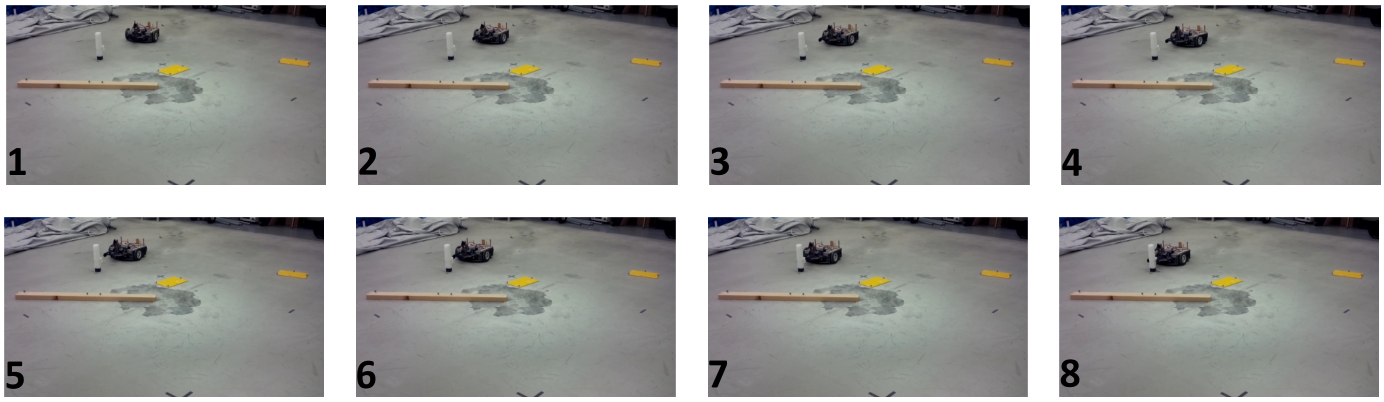


Fig. 12. Clips of video record for UGV going to the goods.

($l_1 \cos 60°$), and from the gripper tip to the vehicle is around 23 cm ($l_1 \cos 60° + l_3$). Given the start position of the UGV and pick-up location of the goods, path for the UGV going to the goods is planned using the RRT-based approach. As no obstacle is laid between the UGV and goods, the UGV start position (RRT tree root) can feasibly "see" the goal (goods), and a straight line path is returned, see Fig. 11. Also as on the figure, waypoints are added onto the path (small green circles). Pure-pursuit controller is used for the UGV path tracking; real-time measurement of the vehicle position/heading is achieved using the OptiTrack system, command signal (encoder input for right/left wheels) is updated at a time step of 0.01 s, which is then sent to the UGV for execution. The UGV is driven to the goods, till the goods falls into the load-range of the gripper (distance between the UGV and goods falls in 8–23 cm). A video record of the overall test has been posted online,[a] several clips of which are listed in Fig. 12, or as on Fig. 13 the tracking result is presented; desired (planned) path is represented by green circles, and vehicle response history is plotted in blue dot line.

When the UGV reaches the goods load range, given the position of the goods and UGV, we first measure and store the distance between the goods and the UGV ($d_L = y_5$ in Fig. 4), then desired deflections of the servos are generated based on the inversed kinematics in Sec. 1. The arms extend, grip the goods, and lift it to the "grip-to-go" configuration, see Fig. 10(b). Note that for this configuration, we still keep the gripper horizontal ($\delta = 0$); the first arm is put to $\delta_1 = 120°$, and the third arm $\delta_3 = 30°$, which yields a distance of roughly 18 cm ($l_1 \cos 120° + l_2 \cos 30° + l_3$) between the (loaded) goods and the UGV body (we put the goods to the tip of the gripper).

After the goods are loaded, the next stage is to transfer the goods to the goal; RRT-based path planning is used. The

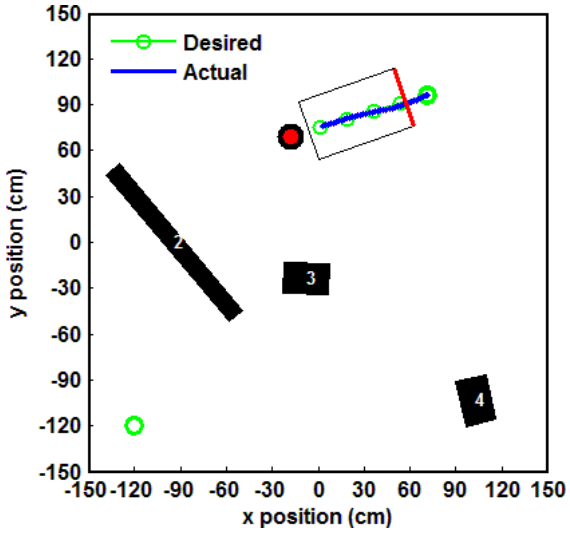
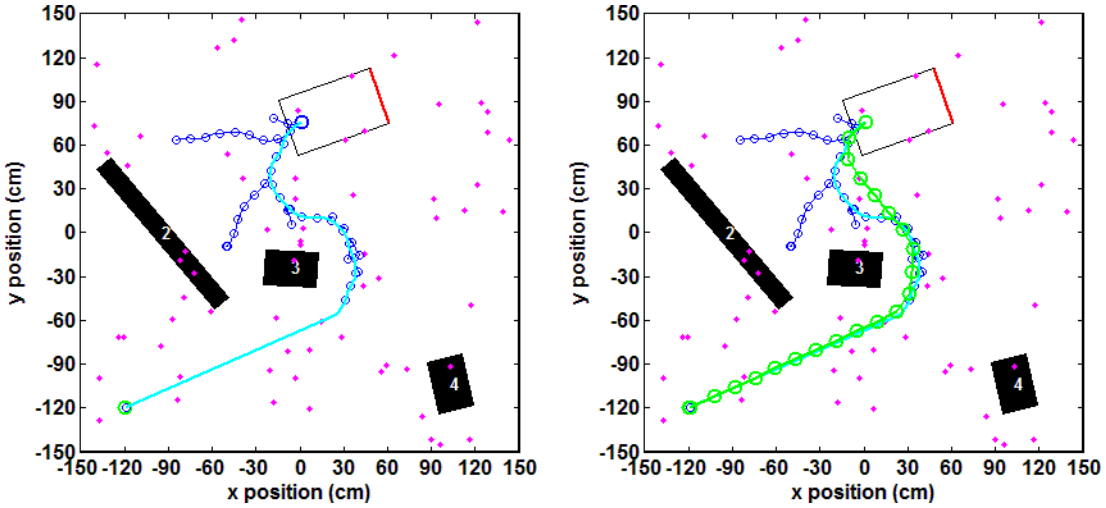


Fig. 13. UGV response history of going to the goods.

RRT tree is expanded to find the path, see Fig. 14 (a); on the figure, magenta dots are the random seeds generated for the tree-growth, and small blue circles on Fig. 14(a) are the nodes that are added to the tree. Final feasible path is connected by tracing from the goal back to the start position (cyan line). After the feasible path is found, optimization is performed, see the green line on Fig. 14(b); waypoints (green circles) are also added to improve the vehicle path-tracking performance. Clips of the video record for this goods-delivering are shown in Fig. 15, or as in Fig. 16 response history of the vehicle is plotted. We note that in the goods-load stage, we store the distance between the goods and the UGV (d_L). When the UGV falls into d_L range of the goal, motion control of the UGV is halted to unload the goods.

The goods unload is a reversed process of loading the goods. The mechanical arms switch from the "grip-to-go" configuration in Fig. 10(b) to the configuration in Fig. 10(a).

[a]This video could be found at: https://www.youtube.com/watch?v = AFzLZ92chDg.



(a) (b)

Fig. 14. (a) RRT tree-grow; (b) planned path after the optimization.

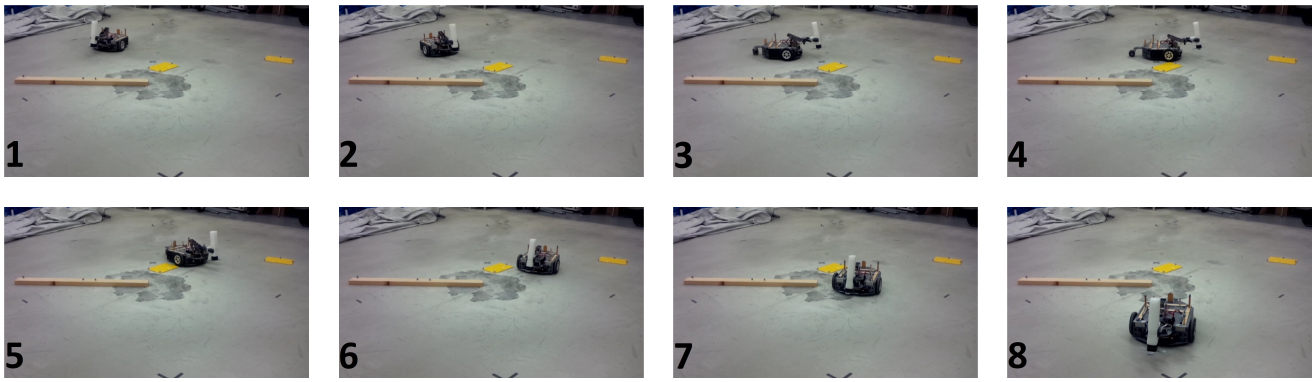


Fig. 15. Clips of video record for UGV transferring the goods.

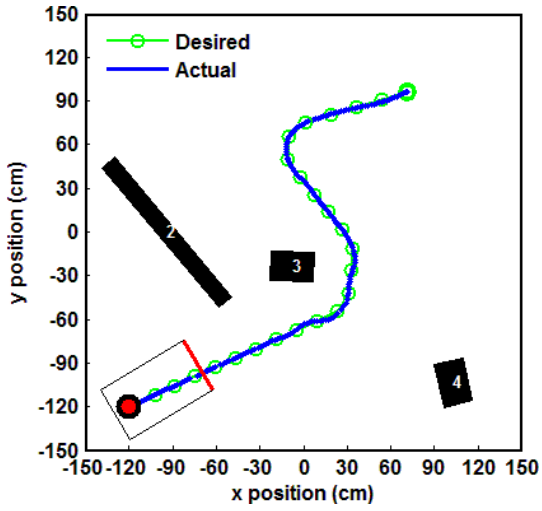


Fig. 16. Overall response history of UGV in the experimental test.

We note that we put the goal at d_L distance from the vehicle, when the arms are reversed to the "go-to-grip" configuration, the goods is put on the goal.

6. Conclusion

This paper presents authors' experimental test of using unmanned ground vehicle for the goods delivery. Configuration, motion equations, and inversed arms kinematics of the discussed UGV is first illustrated. Two drivers for the vehicle motion control are then introduced; advantages/disadvantages of each are discussed. Core part of the goods delivery mission is to find a path which on one hand is collision-free, and on the other applicable for the UGV to execute; path planning in the test is achieved using a RRT-based approach. General structure of RRT algorithm, including tree expansion, node selection, and collision

detection is illustrated; some optimization approach is developed, and waypoints are added onto the path for a better vehicle path-tracking performance.

Final experimental test investigates a scenario which contains four stages: (a) the UGV goes to the goods, (b) pick up the object, (c) transfer it to the goal location, and (d) put it down on the ground. In steps (a) and (c), path connecting the vehicle from the start to goal position is planned using RRT-based approach, which is then delivered to vehicle to execute. In steps (b) and (d), the vehicle halts, and the inversed kinematics for arms manipulation is used to pick-up/put-down the goods. Based on the record presented herein, the experimental test is believed to be a success; proposed framework for the UGV goods-delivery problem in this paper is considered to be applicable.

As a continuation of this work, the authors plan to investigate the problem of UGV goods-delivery in dynamic environment; i.e., dynamical obstacles will be laid within the working space; one foreseeable challenge could be the high computational complexity for the real-time path-planning. Also, certain issues have arisen in the work of this paper, which mainly includes how to generate the tree expansion seeds and how to select from the MCT or pure-pursuit controllers; this could also be one direction of author's future efforts.

Acknowledgments

The author would like thank Mr. Mohammad Ali Askari Hemmat, Mr. Walaaeldin Ghadiry, and Mr. Mohamed Atef Kamel with Department of Mechanical and Industrial Engineering at Concordia University for hardware implementations in the experimental test; the author is also indebted to China's Scholarship Council (CSC) that supports the author as visiting Ph.D. student to Concordia University.

References

[1] K. Ghamry, Y. Dong, M. Kamel and Y. Zhang, Real-time autonomous take-off, tracking and landing of UAV on a moving UGV platform, *Proc. 24th Mediterranean Conf. Control and Automation (MED)*, Athens, Greece, 21–24 June, 2016.

[2] Y. Dong, J. Fu, B. Yu, Y. Zhang, and J. Ai, Position and heading angle control of an unmanned quadrotor helicopter using LQR method, *Proc. 34th Chinese Control Conf. (CCC2015)*, Hangzhou China, July 28–30, 2015.

[3] T. Lozano-Perez, Automatic planning of manipulator transfer movements, *IEEE Trans. Syst. Man Cybern.* **11**(10) (1981) 681–698.

[4] T. Lozano-Perez and M. Wesley, An algorithm for planning collision-free paths among polyhedral obstacles, *Commun. ACM* **22**(10) (1979) 560–570.

[5] H. Edelsbrunner, *Algorithms in Combinatorial Geometry* (Springer-Verlag, Berlin, 1987), pp. 275–278.

[6] P. O'Dunlaing and C. Yap, A retraction method for planning the motion of a disc, *J. Algorithms* **6**(1) (1982) 104–111.

[7] Y. Dong, Y. Zhang and J. Ai, Experimental test of artificial potential field-based automobiles automated perpendicular parking, *Int. J. Vehicular Technol.* **2016**(2016) 2306818, http://dx.doi.org/10.1155/2016/2306818.

[8] Y. Hwang and N. Ahuja, A potential field approach to path planning, *IEEE Trans. Robot. Automat.* **8**(1) (1992) 23–32.

[9] J. Barraquand, B. Langlois and J. Latombe, Numerical potential field technique for robot path planning, *IEEE Trans. Syst. Man Cybern.* **22** (2) (1992) 224–241.

[10] E. Rimon and D. Koditscheck, Exact robot navigation using artificial potential fields, *IEEE Trans. Robot. Automat.* **8**(5) (1992) 501–518.

[11] P. Hart, N. Nilsson and B. Raphael, A formal basis for the heuristic determination of minimum cost paths, *IEEE Trans. Syst. Sci. Cybern.* **4**(2) (1968).

[12] Bryan Stout, The basics of A* for path planning, *Game Programming Gems* (Charles River Media, 2000), pp. 254–263.

[13] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz and S. Thrun, Anytime dynamic A*: An anytime, replanning algorithm, *Proc. Int. Conf. Automated Planning and Scheduling (ICAPS)*, June, 2005.

[14] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp and D. Dolgove, Junior: The Stanford entry in the urban challenge, *J. Field Robot* **25**(9) (2008) 569–597.

[15] S. LaValle and J. Kuffner, Randomized kinodynamic planning, *Proc. 1999 IEEE Int. Conf. Robotics and Automation*, Detroit, MI, USA, May 10–15, 1999.

[16] J. Kuffner and S. LaValle, RRT-Connect: An efficient approach to single-query path planning, *Proc. 2000 IEEE Int. Conf. Robotics and Automation*, San Francisco, CA, USA, April 24–28, 2000.

[17] Y. Dong, C. Fu and E. Kayacan, RRT-based 3D path planning for formation landing of quadrotor UAVs, *Proc. 14th Int. Conf. Control, Automation, Robotics, and Vision (ICARCV2016)*, Phuket, Thailand, 13–15 November, 2016.

[18] Y. Dong, Y. Zhang and J. Ai, Application of RRT algorithm to unmanned ground vehicle motion planning and obstacle avoidance, *Proc. 11th Int. Conf. Intelligent Unmanned Systems (ICIUS 2015)*, Bali, Indonesia, August 26–29, 2015.

[19] M. Kalisiak and M. Panne, Faster motion planning using learned local viability models, *Proc. 2007 IEEE Int. Conf. Robots and Automation*, Roma, April 10–14, 2007.

[20] O. Amidi, C. Thorpe W. Chun and W. Wolfe, (Eds.), Integrated mobile robot control, in *Proc. SPIE*, Boston, MA, Mar. 1991, Vol. 1388, pp. 504–523.

[21] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli and J. How, Real-time motion planning with application to autonomous urban driving, *IEEE Trans. Control Syst. Technol.* **17**(5) 2009 1105–1118.

[22] S. Park, J. Deyst and J. How, Performance and lyapunov stability of a nonlinear path-following guidance method, *J. Guidance, Control Dyn.* **30**(6) (2007) 1718–1728.

[23] Y. Kuwata, J. Teo, S. Karaman, E. Frazzoli and J. How, Motion planning in complex environments using closed-loop prediction, *Proc. AIAA Guidance, Navigation, and Control Conf. Exhibition*, Honolulu, HI, August 2008. AIAA 2008–3257.

[24] A. Yershova, L. Jaillet, T. Simeon and S. LaValle, Dynamic-domain RRTs: Efficient exploration by controlling the sampling domain, *Proc. 2005 IEEE Conf. Robotics and Automation*, Barcelona, Spain, April 2005.

[25] C. Urmson and R. Simmons, Approaches for heuristically biasing RRT growth, *Proc. IEEE 2003 Int. Conf. Intelligent Robots and Systems*, Las Vegas, NV USA, October. 27–31, 2003.

[26] J. Gammell, S. Srinivasa and T. Barfoot, Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an

admissible ellipsoidal heuristic, *IEEE 2014 Int. Conf. Intelligent Robots and Systems*, Chicago, IL USA, September 14–18, 2014.
[27] S. LaValle, *Planning Algorithms* (Cambridge University Press, 2006).
[28] S. Karaman and E. Frazzoli, Sampling-based algorithms for optimal motion planning, *Int. J. Robot. Res.* **30**(7) (2011) 846–894.
[29] P. Jacobs and M. Taix, Efficcient motion planners for nonholonomic mobile robots, *Proc. Int. Workshop on Intelligent Robots and Systems*, Osaka Japan, November 3–5, 1991.