2021 NDIA GROUND VEHICLE SYSTEMS ENGINEERING AND TECHNOLOGY SYMPOSIUM
AUTONOMY, ARTIFICIAL INTELLIGENCE, AND ROBOTICS TECHNICAL SESSION
AUGUST 10-12, 2021 - NOVI, MICHIGAN

Dynamic Object Collision Avoidance for Autonomous Multi-Vehicle Systems in the Robotic Technology Kernel

Matthew Grogan

Southwest Research Institute®, San Antonio, TX

ABSTRACT

As unmanned ground vehicle technology matures and autonomous platforms become more common, such platforms will invariably be in close proximity to one another both in formation and independently. With an increasingly crowded field, the risk of collisions between these platforms grows, and with it the need for path deconfliction. This paper presents two complementary technological developments to this end: a pipeline for affirmatively identifying and classifying dynamic objects, e.g., vehicles or pedestrians; and a pipeline for preventing collisions with such objects. The efficacy of these techniques is demonstrated in simulation, and validation on robotic platforms will be undertaken in the near future.

Citation: Matthew Grogan, "Dynamic Object Collision Avoidance for Autonomous Multi-Vehicle Systems in the Robotic Technology Kernel", In *Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium* (GVSETS), NDIA, Novi, MI, Aug. 13-15, 2019.

1. Introduction

Strategic coordination of multiple robotic systems has long been a goal of the U.S. Military. Such coordination can come in many forms, but the most prominent recent examples of these scenarios come in the form of automated convoys. These have been the focus of several recent U.S. Army projects, including Autonomous Mobility Applique System (AMAS) [1], Autonomous Ground Resupply (AGR), Expedient Leader-Follower (ExLF), and Coalition Assured Autonomous Resupply (CAAR) [2]. Other less-structured scenarios involving multiple platforms independently performing autonomous navigation are also envisioned. To facilitate these missions, two new features, a Dynamic Object Manager (DOM) and a dynamic object collision checking pipeline, have been developed for inclusion in the Robotic Technology Kernel (RTK) [3], the vehicle-agnostic autonomy system of the U.S. Army Combat Capabilities Development Command Ground Vehicle Systems Center (CCDC GVSC).

In convoy settings, autonomous vehicles need to be able to distinguish between fellow convoy members and other entities to respond appropriately. Absent this distinction, convoy members within sensor

range can appear as obstacles like any other. In tightly grouped formations, these neighbors may trigger collision detection algorithms to invoke collision avoidance measures such as applying brakes or changing course, resulting in the undesired disruption or dispersion of the formation. To prevent this disruption by friendly vehicles, an alternative approach is to operate without obstacle detection and avoidance methods, but this approach leaves the vehicle blind to legitimate obstacles. Disambiguating fellow convoy members from other potential obstacles resolves both challenges. In this research, a DOM on each vehicle curates a persistent list of surrounding dynamic objects by fusing sensor input and state information shared between vehicles via radio to produce an accurate model of the convoy dynamics in relation to the ego vehicle. This dynamic object list is used to filter objects identified as convoy members. Traditional obstacle detection and avoidance systems can then operate on the filtered costmap to protect the vehicle as required.

While this first method enables effective collision avoidance in convoy scenarios, the second method reduces conflicts in unstructured multi-vehicle scenarios. In this scenario, an arbitrary number of vehicles operate independently of each other and with unique objectives; e.g., concurrent squad-level resupply and evacuation missions, complex breach, etc. The approach consists of two parts: an aggregator of state information such as position, velocity, and planned trajectory shared between vehicles via radio; and a proactive dynamic object collision checking algorithm for determining potential collisions between vehicles using that state information. While this system is nominally designed for handling vehicle trajectories, any dynamic object with a known trajectory can be treated in a similar manner. In this scheme, each vehicle or object is assigned *a priori* a priority for determining right-of-way when trajectories interact. These right-of-way priorities are currently pre-configured, but the infrastructure for situationally adaptive prioritization is in place. Given shared state, trajectory, and priorities, each vehicle deconflicts its trajectory from those with which it projects it will interact. This deconfliction occurs at the level of path execution; i.e., modifying speed commands, rather than alteration of the path itself. In the absence of truly cooperative path planning, choosing path execution as the locus of control reduces the likelihood of deadlock scenarios by engaging collision avoidance behaviors preemptively. This second technique effectively prevents collisions at the execution layer, but when paired with a real-time path planner that continually updates in response to changes in the world model, stutter-start and leap-frogging behavior may result. This undesirable behavior is mitigated by leveraging the DOM and costmap level dynamic object filtering described above.

2. Background

The RTK is a software ecosystem containing numerous autonomy, perception, and control components. This section will outline several components relevant to the remainder of this paper.

2.1. Path Planners

Within the RTK library, there are various path waypoint navigation methods, with Vaquerito [4] and Maverick [5] being referenced as part of this research effort due to their distinct use cases.

Vaquerito is a path registration technique that is applied to manually constructed paths provided by an operator. It is designed for consistency of path execution and assumes that a given path at least loosely corresponds to a road or trail. Vaquerito operates on one of the available representations of the

physical environment in RTK, two-dimensional traversability arrays commonly referred to as costmaps. Roads and trails have a high traversability and are represented by a low cost, while the opposite is true for off-road areas. Vaquerito compares the operator-supplied path to a costmap of its environment and attempts to register that path to regions of low cost. By doing so, Vaquerito can overcome localization error due to Global Positioning System (GPS) and can adapt inaccurate hand-drawn routes to roads and trails.

Maverick is a real-time path planner, comprising a waypoint graph planner and a Rapidly exploring Random Tree (RRT*) planner, which is capable of rapid response to changes in the surrounding environment. This capability makes it ideal for unstructured and off-road scenarios. Maverick takes in a costmap and ordered waypoints to be achieved and computes the fastest path passing through each waypoint each time the costmap changes. As a probabilistic planner, Maverick-generated paths, even for the same costmap and waypoint input, are rarely reproducible.

Path execution is handled by a separate Path Following Controller (PFC) utilizing the pure pursuit algorithm [6]. The PFC ensures that commanded motion does not exceed the capabilities of the vehicle and provides a collision checking module. The output of the PFC is a predicted path consisting of an array of poses with associated speed and curvature profiles for motion execution.

2.2. World Modeling

In the RTK library, knowledge of the surrounding environment is obtained by sensors such as lidar, radar, or cameras. Costmaps for use in navigation are constructed based on the probability of a given costmap cell being occupied as determined by available sensor data. For dynamic environments, cells must be cleared when an object has moved. In the case of lidar, a commonly used sensor, this cell clearing is achieved by raytracing lidar point returns through occupied cells; i.e., formerly "occupied" cells that can now be seen through are no longer considered obstructed.

3. Dynamic Object Handling

The DOM is a new component of the RTK library. It serves as an aggregator for information about dynamic objects, both from local sensor data and as shared by neighboring vehicles running RTK software.

3.1. Operation

In its current state, the DOM receives information about dynamic objects from two sources: other RTK enabled vehicles and radar. When information sharing is enabled, each system will transmit state information about itself such as platform identifier, position, vehicle dimensions, planned trajectory, etc., which is received by the DOM and used to maintain a persistent list of nearby vehicles. When a platform is equipped with radar, a persistent list of radar detected objects is also maintained. When both lists exist, the radar objects are used to improve the position of corresponding vehicle objects, thereby removing GPS error from transmitted positions. These lists are merged into a single dynamic object list, excluding duplicate objects, and shared with the rest of the system.

3.2. Costmap Filtering

There are two immediate benefits to filtering dynamic objects from costmaps. The first is that it allows the PFC to differentiate between static and dynamic objects and thus enables a more nuanced treatment; this capability will be discussed in subsequent sections. The second is that it can greatly improve the quality of a costmap.

Challenging scene geometries can result in dynamic objects leaving "tails" of occupied

costmap cells where the current process of vacating unoccupied cells is not always entirely effective. As seen in Figure 1, these tails can be particularly problematic for convoy settings where the dynamic object and corresponding tail lie directly in front of the following vehicle. In such cases, the following vehicle could be forced to slow or even come to stop until the occupied cell is cleared, or the vehicle path is altered. This issue can be circumvented at the costmap level by preventing the cells containing known dynamic objects from being marked as occupied in the first place.

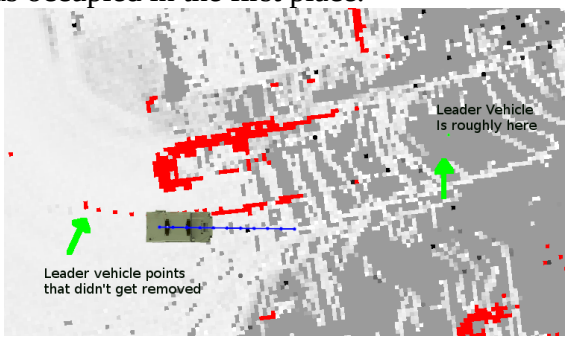

Figure 1. *RTK costmap produced from data recorded during vehicle following tests as an example where the lead vehicle "tail" is clearly visible.*

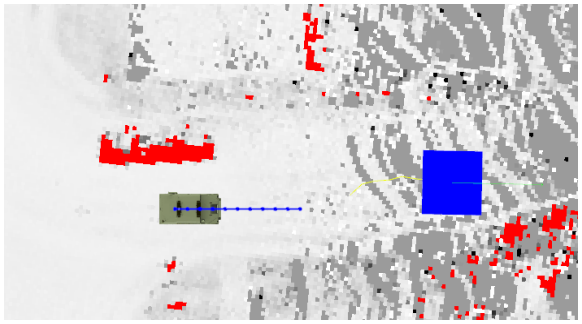

Figure 2. *RTK costmap produced from data recorded during vehicle following tests with lead vehicle, marked by blue square, filtered. Note the absence of an object "tail".*

With this approach, the costmap-generating element can subsequently consume the dynamic object list produced by the DOM and filter all objects which are specified as being fellow members of a convoy as well as any other objects which are not specified as having come to a stop. The results of filtering on example data can be seen in Figure 2, an example costmap which is produced otherwise identically to that of Figure 1 where a dynamic object "tail" is visible.

4. Collision Prevention

The PFC can be configured to handle dynamic object collision prevention through two closely related pipelines. The first uses basic information about objects such as pose (position and orientation), object dimensions, and object speed. The second pipeline utilizes the object's predicted trajectory, consisting of a series of poses with an associated speed profile, as well as an object priority value to determine right of way. Only one of these pipelines will be used per object depending on PFC configuration and available information.

4.1. Basic Collision Check

Dynamic objects are first projected forward using their linear speed to account message latency between DOM and the PFC. Then, for each pose of the predicted path generated by the PFC and for each dynamic object, the following checks are performed.

1. If trajectory collision checking is enabled and the object has an associated trajectory, it can be ignored, as it will be handled by the second collision checking pipeline.
2. A simple radius check is performed to determine if further computation is necessary; if the circumscribing circles of vehicle footprints for both the predicted path pose and the dynamic object pose do not overlap, the object can be ignored.
3. If this point is reached, a collision may be possible and a more refined check utilizing the separating axis theorem [7] is performed. If no potential collision is found, the object can be ignored.
4. If the distance between this object and the vehicle is found to be shorter than

that of any other object, then the object orientation and speed is used to determine if it is currently being followed by the vehicle. When an object is being followed, the commanded speed profile for the predicted path will be adjusted to maintain a speed adaptive, safe distance from the object. Otherwise, the vehicle will halt for the object. The following case is illustrated in Figure 3.

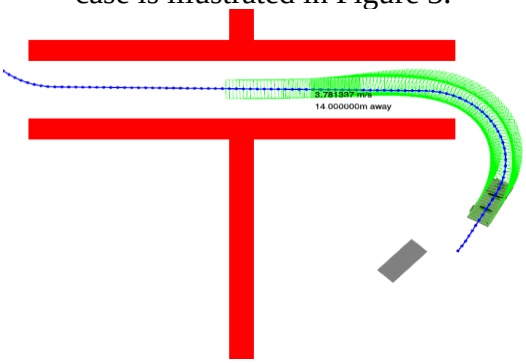

Figure 3. *Basic collision checking involving three simulated vehicles traversing a path (white) between obstacles (red). Here, the ego vehicle, farthest right, can be seen adaptively matching the speed of the lead vehicle, the green rectangle in the tunnel, as it traverses from right to left. The third vehicle, the bottom gray rectangle, follows the ego vehicle.*

4.2. Trajectory Collision Check

The trajectory collision check takes a similar approach to the basic check with a few key differences. The following steps are only performed for dynamic objects with corresponding priority values greater than that of the vehicle. First, the dynamic object is projected forward along its trajectory to account for any latency. The combined length of the vehicle and object predicted trajectories are then compared with the physical distance between their current positions; if this distance is greater than the sum of their path lengths, no collision is possible. Next, for each pose of the predicted path generated by the PFC and for each pose of the dynamic object trajectory, the following checks are performed:

1. Using the speed profiles associated with the trajectories, an estimated time of occupation for each pose is calculated. If the difference between these values is greater than pre-configured time window, the dynamic object pose can be ignored.
2. A radius check and a subsequent separating axis check are used as in the basic collision checking approach.

If a potential collision is found, the speed profile for the predicted path is recalculated.

5. Performance Evaluation in Simulation

Due to the challenges of evaluating convoy performance in a pure simulation environment, this section will primarily focus on the collision mitigation aspect of this work with costmap filtering emulated.

Two scenarios are examined, each involving three platforms. The first, partially shown in Figure 4, tests Vaquerito with three routes of approximately equal length through simulated trails intersecting at the same distance in a T-junction. The second, partially shown in Figure 3, tests the Maverick planner with three waypoints configured such that the vehicles must pass through narrow bottlenecks. Scenarios of this nature are selected because they have proven challenging in the absence of the techniques described in this paper. In the Vaquerito case, this is due to orthogonal and head-on intersections which can easily result in deadlock or collision. In the Maverick case, this is due to narrow gaps and passages which, when temporarily obstructed by an unfiltered vehicle, result in highly suboptimal updates to the planned path. Tests are limited to three vehicles per scenario because this configuration is typical of current field testing numbers and because it is sufficient for demonstrating more than the most basic of interactions between vehicles.

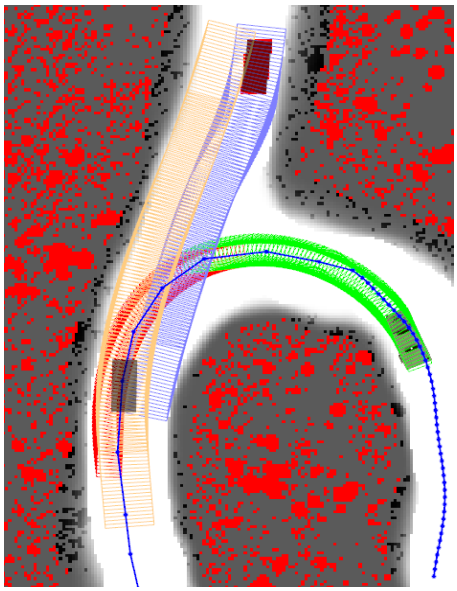

Figure 4. *Trajectory collision checking. Higher and lower priority paths are represented by the orange and blue path footprints respectively. The estimated collision point along the predicted path is indicated by the transition from green to red.*

The shortest distance required for each vehicle in both scenarios is approximately 120 m, and the speed limit has been set to 5 m/s. The metrics used for evaluation are the time required for mission completion and the number of deadlocks or collisions observed. For both scenarios, twenty iterations of the following feature configurations are tested:

- Case 1: trajectory collision check with costmap filtering.
- Case 2: trajectory collision check without costmap filtering.
- Case 3: basic collision check with costmap filtering.
- Case 4: all dynamic object handling disabled.

Multiple permutations of priority and positions were tested; since overall performance was largely similar across permutations and for the sake of brevity, only one permutation for each scenario is reported. The selected Maverick permutation is the most adversarial, requiring that the high-priority vehicle cross paths with the lower-priority vehicles before reaching the bottlenecks. The distinction between the Vaquerito route permutations is marginal, so the data presented here is representative of the set as a whole.

5.1. Vaquerito Results

For completed routes with recorded execution times, Figure 5 shows that feature configurations with trajectory checking disabled appear to complete slightly faster. This belies the fact that these test cases experienced more failures as shown in Table 1. Basic dynamic object checking is sufficient to prevent collisions but is unable to avoid deadlock situations in which each vehicle is stopped for another. With dynamic object handling disabled entirely, vehicles will operate at higher speeds in proximity, often narrowly missing one another. In several instances this resulted in collisions, highlighted in red.

Table 1*. Observed failure cases, highlighted, for Vaquerito scenario.*

Feature configurations		Case 1	Case 2	Case 3	Case 4
Vehicle 1	# Collisions	0	0	0	2
	# Deadlocks	0	0	4	0
Vehicle 2	# Collisions	0	0	0	3
	# Deadlocks	0	0	6	0
Vehicle 3	# Collisions	0	0	0	1
	# Deadlocks	0	0	6	0

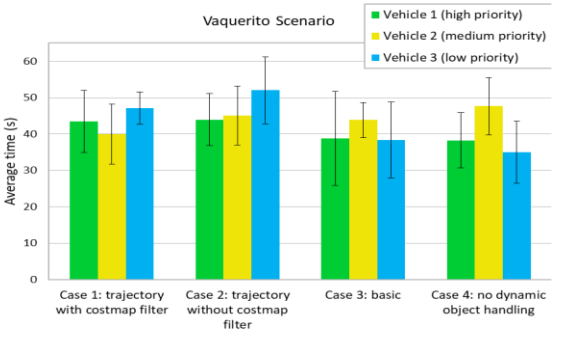

Figure 5*. Average times for Vaquerito route completion, per vehicle. Bars indicate standard deviation.*

The trajectory collision checking test cases are comparable in average route execution time with a moderate performance gain with

costmap filtering. Neither experienced failures. This is to be expected as Vaquerito is designed to register a given route to a region of high traversability rather than actively reroute around dynamic obstacles, so the resultant paths will not be dramatically different. The longer execution times and larger variances seen with the unfiltered costmap are a consequence of the unfiltered dynamic objects affecting Vaquerito's registration method, causing routes to be offset away from the center and toward the higher cost edge of the simulated trails. From these results, it appears that dynamic object trajectory checking with costmap filtering should be the preferred configuration for Vaquerito route following.

5.2. Maverick Results

As with the Vaquerito test results, failure cases are excluded from the average mission completion times. Table 2 illustrates that any level of dynamic object handling is sufficient to prevent unbreakable deadlock and collisions. However, as shown in Figure 6, there is a pronounced decrease in average execution times when costmap filtering is employed. This is a result of Maverick's continuous response to changes in the costmap and the fact that this scenario requires vehicles to pass through multiple narrow passages. Without costmap filtering, whenever a vehicle enters and obstructs such a passage, each other vehicle with a planned path through the passage will immediately compute an alternative, suboptimal path to its next waypoint. While executing these alternative paths, vehicles may end up obstructing yet more passages causing an extended delay in mission execution due to suboptimal replanning. Referring to Figure 3, such a situation might occur when two vehicles enter the narrow passage in opposite directions; until one of the vehicles commits to an alternative route avoiding said passage, no progress will be made. This process can repeat several times before all vehicles successfully execute their missions. Notably, dynamic object trajectory checking exacerbates this behavior by artificially obstructing gaps prior to actual obstruction by a vehicle, thereby prolonging the previously described process.

Table 2*. Observed failure cases, highlighted, for Maverick scenario.*

Feature configurations		Case 1	Case 2	Case 3	Case 4
Vehicle 1	# Collisions	0	0	0	2
	# Deadlocks	0	0	0	0
Vehicle 2	# Collisions	0	0	0	10
	# Deadlocks	0	0	0	0
Vehicle 3	# Collisions	0	0	0	8
	# Deadlocks	0	0	0	0

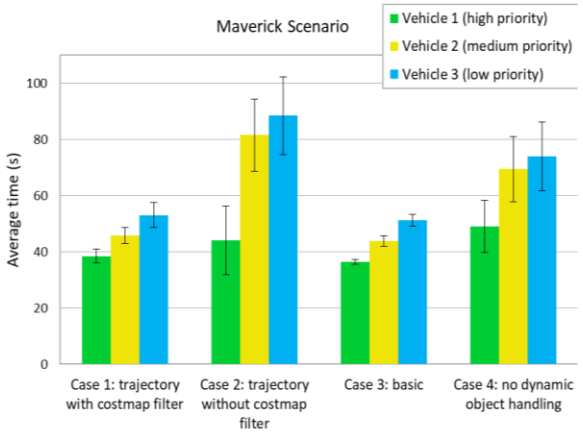

Figure 6*. Average times for Maverick mission completion, per vehicle. Bars indicate standard deviation.*

In this scenario, basic dynamic object checking and trajectory checking with costmap filtering have nearly identical performance, trajectory checking having slightly more variance in execution time. Though omitted from Figure 6, permutations of priority and starting position have comparable results with minor differences in average execution time owing to the time required for vehicles to order themselves by priority before passing through passages.

These results along with those of Section 5.1 indicate that dynamic object trajectory checking with costmap filtering may be effective at decreasing path execution time and preventing deadlock and vehicle collisions across a range of use cases. With larger numbers of participating vehicles, the

patterns seen here are expected to become more pronounced; without dynamic object handling features, mission execution times will rise due to the increased likelihood of adverse vehicle interactions and failure cases such as collisions become all the more probable.

6. Future Work

This paper has presented preliminary results based on simulation for dynamic object handling and collision mitigation, and validation on vehicles is ongoing. Radio bandwidth usage and computational overhead for large numbers of dynamic objects in the field should be profiled to determine whether and when increasing the number of vehicles decreases system performance. Persistent dynamic object tracking using lidar and vision data has also yet to be implemented with the proposed approach to handling dynamic objects.

A natural extension of this work is dynamic assignment of vehicle prioritization. One likely approach to this is a rule-based encoding of human knowledge such as giving way to oncoming traffic or yielding priority to leading vehicles. Another would be assignment of priority based on some measure of the utility of a vehicle's current trajectory relative to alternative trajectories generated in response to the presence of other nearby vehicles; those with trajectories subject to the largest reduction in utility may be given precedence over others. Prioritization based on utility is closely related to a second planned extension of this work, the elevation of path deconfliction from the motion execution layer to the path planning layer. By applying similar principles during path planning rather than motion execution alone, efficiency of the system as a whole can be improved by preventing conflicts at an earlier stage.

7. Conclusion

The two techniques described here complement each other to provide a straightforward and effective solution to effectively navigating friendly multi-vehicle maneuvers in both structured (formation) and unstructured (independent) scenarios. The proposed Dynamic Object Manager maintains a list of radio-connected dynamic objects, and the developed dynamic object collision detection algorithms provide the capability for avoiding collisions with such objects. Identified vehicles are filtered at the costmap level, resulting in improved costmap quality and path stability and, since the dynamic object collision detection algorithms operate independent of the costmap and world model, any previously existing safety functions at that level are not compromised. Recorded real-world data and simulation experiments have indicated that the proposed approach can be effective in handling dynamic objects. These two capabilities serve as important enabling technologies for future developments in autonomous multi-vehicle systems.

8. REFERENCES

[1] Bernard Theisen, Autonomous Mobility Applique System (AMAS) JCTD, Jun 23, 2011, https://apps.dtic.mil/sti/pdfs/ADA550916.pdf Powerpoint Presentation.

[2] D. Pirozzo, J.P. Hecker, A. Dickinson, T. Schulteis, J. Ratowski, and B. Theisen, "Integration of the Autonomous Mobility Appliqué System into the Robotic Technology Kernel", In Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium (GVSETS), NDIA, Novi, MI, Aug. 13-15, 2019.

[3] Dr. Robert Kania; Mr. Phil Frederick; Mr. William Pritchett; Mr. Brian Wood; Mr. Chris Mentzer; Dr. Elliot Johnson "Dismounted Soldier Autonomy Tools (DSAT) - From Conception to

Deployment", In Proceedings of the Ground Vehicle Systems and Technology Symposium (GVSETS), NDIA, Novi, MI, Aug. 12-14, 2014.

[4] N. Alton, M. Bries, J. Hernandez, "Autonomous Convoy Operations in the Robotic Technology Kernel (RTK)", In Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium (GVSETS), NDIA, Novi, MI, Aug. 11–13, 2020.

[5] N. Seegmiller, J. Gassaway, E. Johnson and J. Towler, "The Maverick planner: An efficient hierarchical planner for autonomous vehicles in unstructured environments," *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 2018-2023, doi: 10.1109/IROS.2017.8206021.

[6] R.C. Coulter, "Implementation of the pure pursuit path tracking algorithm", Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, 1992.

[7] S. Boyd, L. Vandenberghe. "Convex optimization", Cambridge university press, 2004.