Autonomous Robotic Convoy System Design

CSCI 5551 Final Project

Ming-Chang Chiu, Wenbo Dong, Joel Runnels, Venkat Ram Subramanian

I. Abstract

In this project, an autonomous robotic convoy system based on robots capable of measuring distance as a function of angle was developed. The convoy system was designed to follow an arbitrary leader along a path, while avoiding obstacles. The algorithms developed for this project have proven reasonably successful in a variety of cases, although the software is not fully robust to all scenarios, and still malfunctions under some conditions.

II. Introduction

Multiple trailers pulled by one truck, sometimes called "road-trains" are a common and efficient method of transporting freight on long hauls. Road-trains offer the advantages of decreased aerodynamic drag and more freight carried per driver. However, they are accident prone and often difficult to control, and are thus limited in their length and use. The development of a robotic vehicle convoying system, in which each vehicle in the unit follows the next unit without a mechanical linkage, would theoretically offer the same benefits of a road train system without the difficulties associated with mechanical coupling between cars. In this report, we present the results of our efforts towards developing such a robotic convoy system. Although the problem addressed here is admittedly a simplified version of the actual problem, the methods and algorithms developed here could be used as the basis for a more complex convoy system capable of performing in a wider range of conditions.

III. Literature Review

Article [1] deals with vehicle platooning, in which a convoy aims at following its leader's path closely and safely with neither collision nor lateral deviation. The authors propose a platooning algorithm based on a near-to-near decentralized approach. Each vehicle estimates and memorizes online the path of its predecessor as a set of points. After choosing a suitable position to aim for, the follower estimates online the predecessor's path curvature around the selected target. Based on a heuristic search, it computes an angular velocity using the estimated curvature. The optimization criterion allows the robot to follow its predecessor's path without oscillation while reducing the lateral and angular errors.

From article [2], we learn that various "Urban Transportation Systems" are currently in developing, in order to put forward solutions to congestion and pollution in dense areas. Autonomous electric vehicles in free-access can be seen as an attractive approach, in view of the large flexibility that can be expected. The author proposed a solution for this named platoon motion: several autonomous vehicles accurately follow the trajectory of a manually driven first vehicle, with pre-specified inter-distances. Also a global decentralized platoon control strategy, supported by inter-vehicle communications and relying on nonlinear control techniques is proposed. Each vehicle is controlled with respect to the same smooth reference trajectory, inferred on-line from the motion of the first vehicle via B-spline optimization.

In article [3], the authors have categorized and discussed the Vehicle Platooning literature published between 1994 and 2010. The article classifies the platooning literature into two broad divisions: application based and theory based. From the categorization of application based results, we found that platooning applications

are mainly related to mobile robots, military, agriculture, underwater vehicles and aerospace. From the survey of theory focused results, it is seen that platooning theory focuses broadly on inter-vehicle communication techniques in platoons, obstacle detection and collision avoidance techniques, lateral and longitudinal control strategies for platoons, trajectory planning methods and achieving string stability for a platoon. The concept of platooning is a relatively new one. Furthermore, introduction of vehicle platooning without modifying existing infrastructure poses a challenge.

IV. Problem Description

This project addresses a basic robotic convoy system. Specifically, our project addresses a convoy of robots arranged one after another. The robots used in this project are the Pioneer 3 robots made by Adept Mobile Robots, equipped with SICK laser scanners. The Pioneer 3 robots are tricycle robots with two drive wheels and one trailing wheel, while the SICK laser scanners measure distance as a function of angle from zero to 180°. In this project, the first robot in the convoy follows an arbitrary leader along an arbitrary path, while the rest of the units in the convoy are specifically programmed to follow the next robot in the convoy, rather than an arbitrary object. In the testing of this project, a person walking in front of the convoy served as the leader, but in general the leader could be an arbitrary object, as long as it is sufficiently large enough to be detected by the robots.

In our treatment of the robotic convoy system, multiple assumptions were made. A basic starting configuration is assumed for the convoy, specifically, the follower units of the convoy are assumed to be positioned such that the leader (or next robot in line) is the closest object in the follower's field of view, while the leader is assumed to have an arbitrary position. Some assumptions were made out of necessity due to the physical and hardware limitations of the robots used. Specifically, the convoy software assumes that the leader will never be more than three meters away, and that the leader will not traverse a path which is physically impossible for the robots to follow. The algorithm assumes that there will be obstacles present in the robot's workspace, but it assumes that any obstacles are fully visible by the robots (i.e. there are no obstacles which are too high or too low for the robot to see.) Similarly, the algorithm assumes that all obstacles are detectable by the laser scanner, meaning that no obstacles are highly reflective or transparent.

V. Problem Solution

In order to effectively address this problem, the general problem of following a leader while avoiding obstacles was broken down into multiple parts. The general workflow of the robot is specified as follows. After power-up, the robot continuously scans for objects, looking for a suitable target to follow. Upon detection of such a target, the robot plans and executes a path towards that target. As the robot is executing this maneuver, it continuously updates the position of the target, and alters the planned path accordingly. If the target is lost by the robot, then the robot stops and either begins looking for a new target, or simply reentering rest mode, depending upon whether the robot is the first unit in the convoy. These parts and their treatment in this project are detailed below.

A. Object detection

In order to follow an object, the robots must first be able to detect the presence of objects. In order to accomplish this task, a simple object detection algorithm was developed. The basic method for detecting objects is to look for large discontinuities of the value of range between angles. However, because the workspace is not always clean, there can be significant background noise in the laser scan data, which

can lead to large discontinuities simply due to noise rather than the presence of an object. To avoid this issue, the software uses the natural log of range value to scan for objects. The natural log operation has the effect of amplifying the discontinuities in range due to near-field objects of interest, while filtering long-range noise.

As stated above, the software identifies objects by searching for discontinuities in the depth data, which signify edges. With this approach, there are two types of edges; those which signify the start of a near field object, which the code calls "right edges," and those which signify the end of a near-field object, which the code calls "left edges." Right edges are indicated by a large decrease in range, while left edges are indicated by a large increase in range. The software begins by searching for potential right edges by looking for an angle where the decrease in natural log of range is sufficiently large to indicate an object edge. Once a right edge has been found, the software continues through the range of angles looking for either a left edge, or another right edge. If a left edge is found, then the software stores the left and right edges as the boundaries of an object within the robot's field of view. If another right edge is found, then the robot discards the first right edge, and continues looking for a left edge of an object. In this way, the software only detects as an object those objects that are fully visible to the robot and which have no major discontinuities between their edges.

Depending on what stage the robot is in, the robot may or may not need the velocity of the objects spotted. If velocity is not required, the software simply returns a list of objects and their associated information, including distance, degree, and left and right edges. If velocity is required, then the software performs two object detection scans, separated by a certain time specified by the calling function (usually 0.1-0.2 seconds). The software then compares the position of each object in the first scan to the position of each object in the second scan. For each object pair, the software calculates what the object's velocity would be in order to travel between those two points in the amount of time between scans. Whichever object pair yields the lowest velocity is assumed to be the correct matching. In this way, the software is able to calculate the position and velocity of multiple objects in the range of view of the laser scanner.

B. Target selection

The robot initially performs continuous scans for objects, looking for a suitable target to follow. The selection criteria for a target depend upon the robot's position in the convoy. For all the units except the first in the convoy, the robot simply selects the closest object in the field of view as the target object. This simplified selection method is allowable because of the assumed starting configuration of the convoy.

The first robot in the convoy does not pick the closest object as its target, but rather selects the closest object which is moving within a certain velocity range. Because velocity measurements can be inaccurate due to noise in the measurements, the robot serving as the first unit in the convoy always performs an additional scan after detecting a moving object, to verify that the object in question is in fact still moving at the velocity previously indicated.

Regardless of the robot's position in the convoy, once the target is selected, the robot stores the location and size of the target, then enters path planning mode.

C. Path planning

Path planning consists of two basic parts: speed selection and heading selection.

Speed selection: Setting the correct speed is of upmost importance for the robot to function correctly. Consequently, a speed control scheme which guarantees zero steady-state error is

highly desirable. In order to ensure zero steady-state error, an integral control scheme was implemented in this software. This design decision was motivated by the fact that a system with integral feedback has zero steady-state error, provided the system is stable. [4] The speed control scheme used in this software can be written as follows:

$$|v| = k_f \left| \int_0^t (v_{target} - v_{robot})dt - r_{min} \right| \quad (1)$$

where r_{min} is a constant. Since the integral in (1) is simply the difference between the robot's position and the target's position, (1) can be rewritten as:

$$|v| = \left(d_{target-robot} - r_{min}\right)k_f \quad (2)$$

This is the expression implemented in the code. k_f and r_{min} are both variables which the operator may adjust at runtime, allowing the operator to tune the control scheme to achieve desired response characteristics.

After calculating the speed based on integral methods, the software has a number of "safety checks" to ensure that the calculated speed is in fact safe. Among these are a limit on speed based on how close the robot is to *any* object (if the robot is navigating around close-proximity obstacles, it should not be doing so at a high speed), and checks to ensure that the speed is not higher than a user-selectable maximum speed.

Heading selection: By default, heading is controlled in a proportional manner. Specifically, at each iteration, the robot calculates a desired change in heading equal to 25% of the difference between the robot's current heading and the target object's position. This proportionally determined heading becomes the robots "desired" heading, but is not executed until the robot can determine whether it is a safe heading.

In order to determine whether or not a heading is safe, the robot performs another laser scan. The robot then transforms these data from the original coordinate system (x_o, y_o) into the coordinate system which the robot *would* have after performing the desired heading change (*x'*, *y'*). For each point in the scan, the robot projects that point into the (*x'*, *y'*) coordinate system. It then checks to see if the measured range causes the robot's desired path to be unsafe. In order to determine whether a measurement is problematic, the robot first compares the data point's *y'* coordinate to the distance between the robot and the target. If the point's *y'* coordinate is greater than the distance to the target, then the robot can ignore this point, as the robot would not encounter this point before encountering the target. If, however, the point's *y'* coordinate is less than the distance to the target, the robot checks to see if the absolute value of the point's *x'* coordinate is greater than the robot's width plus a safety margin. If the point's *x'* coordinate falls outside of that safety margin, then the point is considered to be not problematic. However if the point's *x'* coordinate falls inside the safety margin, then it is an unsafe point, and the path is not safe for the robot to follow. The criteria for a point to be determined as unsafe are expressed below:

$$\begin{aligned} d * \sin(\theta - \delta) &> l_p \\ d * \cos(\theta - \delta) &> (w_r + sf)/2 \end{aligned} \quad (2)$$

where d is the distance measured at angle θ, δ is the desired change in heading, l_p is the length of the path to be traveled, w_r is the width of the robot, and sf is a width safety factor. Both conditions in (2) must be met in order for a path to be determined unsafe. These conditions are illustrated in Figure 1.

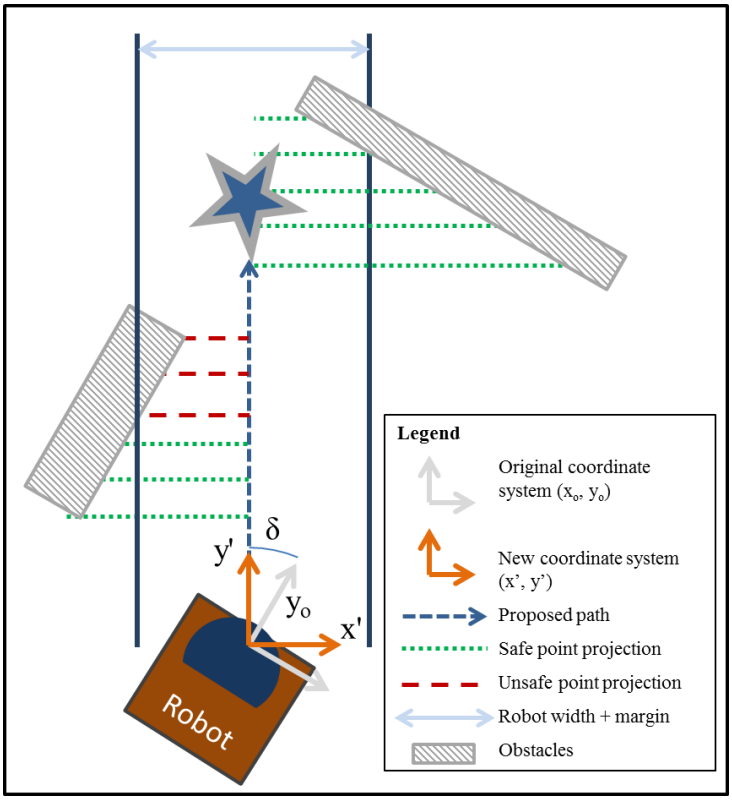


Figure 1: Path safety determination procedure

Upon the detection of the first unsafe point, the software throws out this path as a possible path. The robot then alters its proposed path by plus or minus one degree (depending on where the obstacle was detected), and begins checking that path for safety. The robot repeats this iterative process until a safe path is found.

Upon the determination of a safe velocity and safe heading, the robot issues the command to execute that path. Once the heading change maneuver has been completed, the robot updates its stored position of where the target object "should" be.

D. Target position updating

After setting the heading and velocity as calculated in the path planning stage, the robot does another scan for objects. The robot then compares each object in the scan to the position of the previous target object, to try and determine where the target is at the next instant in time. The software also compares the size of the new objects to the size of the last target object. If an object is found in the new scan which is sufficiently similar (both in size and position) to the previous target, then the robot stores this object as the new target. If no match is found, then the robot instead performs an estimate on where it thinks the target "should" be, based on the robot's velocity since the last path update, the amount of time elapsed since the last update, and the robot's change in heading. The robot then enters the path planning phase again to plan a new path, using either the updated target to plan the path, or else using the estimated position of the target based on robot speed and velocity.

Finally, if the robot has determined that it has reached its target and the target is not moving, then it stops and waits for the object to move. If the robot is the first robot in the convoy, then it will wait five seconds, and if the target has not moved for that length of time then the robot begins searching for a new target. If the robot is not the first unit in the convoy, then the robot will continue waiting indefinitely until it detects that its target has moved. The overall algorithm of the robot is shown in Figure 2.

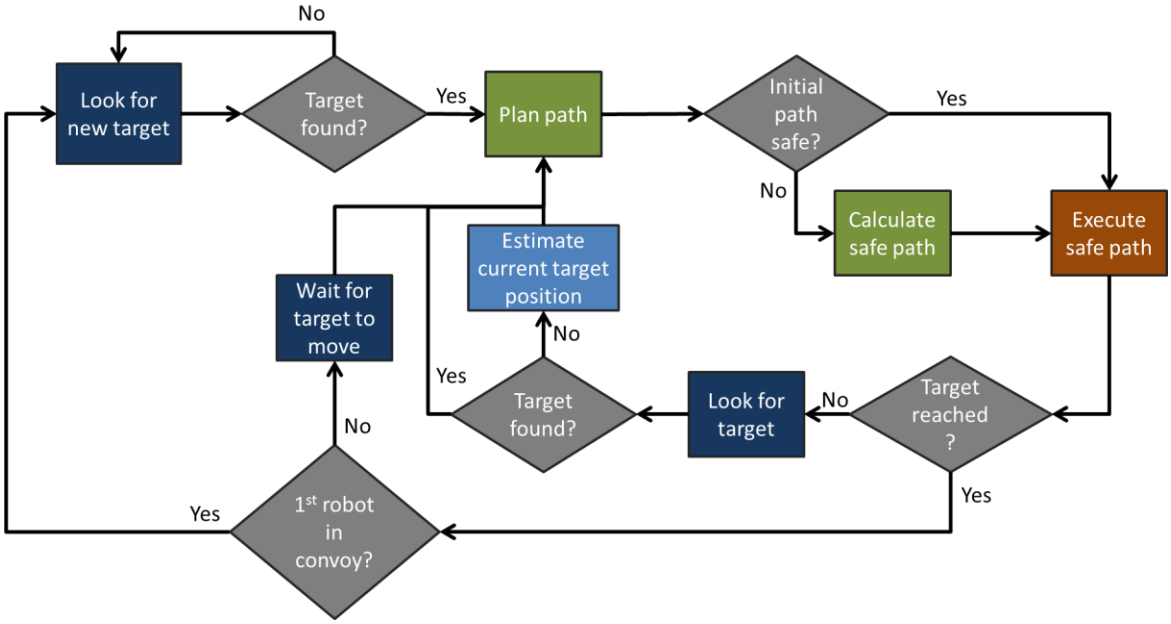


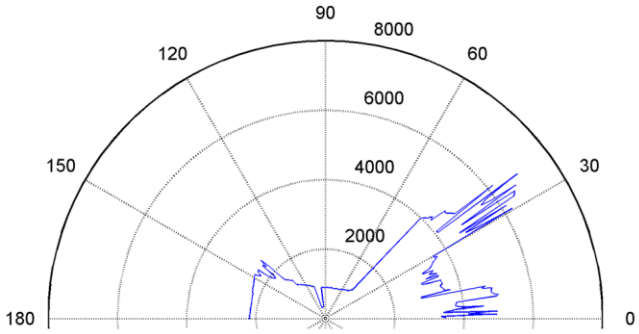
Figure 2: Object following algorithm flowchart

VI. Results

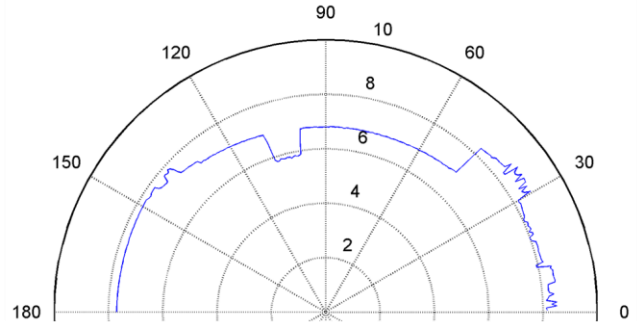
A. Object detection

The algorithms described above have proven to be very reliable for the detection of objects. The use of natural logs for detecting change in range has worked very reliably at filtering out large amounts of far-field noise, while consistently detecting objects in the robot's near range.

The robot's velocity measurement system is somewhat less reliable, due to the noise in the measurements of position which result in high amplitude noise in the actual velocity measurements. This problem is addressed by taking multiple readings to confirm the object's velocity, and while this solution works acceptably well, it is not entirely reliable, and often results in the robot failing to detect objects with real velocities.



a) Distance as a function of angle. Near field object at ~105° is visible, but relatively small compared to far-field noise.



b) Natural log of distance as a function of angle. Here, the near-field object ~105° is much more obvious, while the far-field noise is minimized.

Figure 3: Sample data set displaying effectiveness of natural log for far-range noise filtering

B. Target selection

The target selection algorithm is fundamentally tied to the object detection algorithm, and can only function as well as the object detection system. For the units in the convoy which are not the first unit, the system is very effective at identifying a target, since this method of target selection does not depend on a measurement of the target's velocity. However, for the first unit in the convoy, target selection is somewhat less reliable. This is due to the heavy filtering applied to velocity measurements, which in turn filters out some real velocities. The level of filtering was an intentional design decision, since it was determined that it is safer for the robot to not follow a real target than for the robot to chase after targets which are actually not moving.

C. Path planning

The path planning algorithms have proven to function very well in a variety of circumstances. The robot is effectively able to navigate around a wide range of obstacles including walls, chair and table legs, and humans in the robot's path. Since the robot continuously updates the safety of the path, it is able to actively avoid both moving and stationary objects with a high degree of success. In testing this software, the only times the path planning algorithm has failed is when no safe path exists for the robot, or when an obstacle is invisible to the laser scanner.

D. Target position updating

The target position estimation and updating works with an acceptable amount of success, although there is room for improvement. The robot is fully capable of handling cases where the target temporarily disappears from view, whether behind a wall or simply because the robot failed to detect the object on a particular scan. However, it is still quite possible to "trick" the robot by intentionally performing maneuvers that are too fast for the robot to accurately estimate the target's new position. The robot can also be fooled if the target it is following disappears from view behind a wall and then moves a sufficiently great distance away from its last position, causing the robot to lose the target altogether.

E. Overall Results

Overall, the robot's algorithm functions very well under a wide variety of conditions. Under "ideal" conditions (target visible most of the time, no sudden maneuvers that cause the followers to lose their targets), the convoy unit is very consistent and reliable. Under suboptimal conditions (no safe path between obstacles, rapidly moving, unpredictable target, sudden sharp maneuvers), the convoy system's performance is less predictable. In some cases the convoy will still perform as expected, however, as stated above, it is possible to fool the system into losing its targets.

Additionally, despite best efforts at refining the code, there are still various subtle bugs and errors which occasionally lead to anomalous behavior in the robots, as would be expected for a development code which has not been fully vetted.

VII. Conclusions and Future Work

In this project, a simple robotic convoy system was developed in which the units reliably track a target object, in a variety of circumstances. This system has proven to be robust against a numerous challenges, including navigation around stationary and moving obstacles, and uninterrupted following of the target even with temporary loss of target visibility. Immediate future work would include further refinement of the object detection and velocity measurement algorithm, as well as expanding the capabilities of the object's target position estimation system.

Other future work could include the development of a more refined velocity control scheme that would enable the robots to safely maintain a smaller following distance even at high speeds. Long-term future work could include the use of transmitters between units to relay commands, and absolute position measurement (GPS) to allow the follower units to relocate the leader in the case where they lose the target completely.

VIII. References

[1] J. Yazbeck, A. Scheuer and F. Charpillet, "Decentralized Near-to-Near Approach for Vehicle Platooning based on Memorization and Heuristic Search," in *IEEE International Conference on Robotics & Automation*, Hong Kong, 2014, pp.631-638

[2] P. Avanzini, B. Thuilot, T. Dallej, P. Martinet and J.P. Derutin, "On-line reference trajectory generation for manually convoying a platoon of automatic urban vehicles," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis, USA, 2009, pp. 1867-1872

[3] P. Kavathekar, Y.Q. Chen, "Vehicle Platooning: A Brief Survey And Categorization," in *ASME 2011 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, Washington, DC, USA, 2011, pp.829-845

[4] Dorf, Bishop, Modern Control Systems, New Jersey: Pearson, 2011, pp.250-253