Automated convoying of trucks utilizing waypoints

Alexandru Cosmin, Anders Egelund Kjeldal, Jesper H. Jørgensen,
Jonas Ørndrup, Nikolaos Biniakos and Tomas Tijunaitis
Section of Control and Automation
Institute of Electronic Systems
Aalborg University, 9220 Aalborg Øst, Denmark

***Abstract*—The Danish Military has requested for research in platooning of military trucks, such that the whole platoon can be driven by one driver. The driver is to be placed at random in any truck that is not the first one, to ensure safety. The driver then remote controls the first truck, while the other trucks should follow the tracks of the first truck autonomously.**

Two Turtlebots are used as proof of concept testing for the problem. A system design is made such that the first Turtlebot is remote controlled via a PS3 controller and the second Turtlebot follows the tracks of the first one. The two Turtlebots are limited such that they steer similar to trucks, in order to make it more comparable. In order to be able to follow the first Turtlebot, it sends waypoints, consisting of its position and heading, to the second one.

Two different control designs to follow the route of the first Turtlebot is used, in order to find the better one. One of the designs assumes that all paths between waypoints can be approximated with one curve, where the parameters to obtain the curve is found and set to the motors. The other design assumes that the waypoints are dense enough to take straight lines between them. The two designs are compared with different waypoint frequencies, where the mean square error is used to determine, which controller is the better at the different frequencies.

I. Introduction

This paper is based upon a wish from the Danish Military to platoon military trucks. Platooning of military trucks is a wanted feature, because transportation of military goods often happens by truck, where multiple trucks drives together to increase efficiency and safety. It is also wanted to spare manpower in this process and thereby have some of the vehicles autonomously following each other. Another aspect of the project is safety. To protect the driver from attacks, the driver cannot be placed in the front vehicle, as this is the vehicle most often hit by attacks. Therefore the driver must be able to control the platoon, while being placed in any of the vehicles. Additionally the driver must not be placed in the same truck every time, as this would allow the enemy to plan for this and attack the vehicle with the driver. A solution for this could be the driver controls the front truck directly based on a camera feed, and thereby can be seated in any of the trucks or even at a remote location. This solution is the foundation for this paper.

Other research have been done in this area, however most of these deal with driving on established infrastructure, such as highways. The approach of these methods is to track the vehicle position with respect to the road, such that they stay in the correct lane [1]. This approach is not possible in this project, as the scope of this project is to design a solution that is usable in situations where the road conditions are poor to non existing.

To circumvent the lack of a road, this project uses waypoints to establish a frame of reference, which uses global coordinates instead. As the project is subject to time limitations, it is not feasible to research the project at full scale. Therefore two Turtlebots, referred to as TB1 and TB2, are used to emulate the trucks and making this paper a proof of concept. In order to avoid placing the driver in the first vehicle, TB1 will be controlled by an external driver, and the subsequent Turtlebots needs to follow the tracks of the first as precisely as possible. In this project all the robots will be assumed identical, and driving on good condition surface, meaning no slip of the wheels. A communication is set up between the two Turtlebot such that TB1 sends way points to TB2. This project compares the performance of two methods of off-road platooning, evaluating their ability to reduce the deviation from the tracks. In this paper a proof of concept test setup is designed such that TB2 follows in the tracks of TB1, based on the data communicated between them.

II. Methods

A. System concept

The front TB, TB1, is remote controlled and the subject for this paper is to make the second TB, TB2, drives as precise in the tracks of the first one as possible. In order for the second TB to drive in the same tracks as the first one, it needs to know the route driven by the front TB.

The trucks will often drive on gravel roads or in terrain with no roads, because of this, the TBs cannot rely on following the road like most autonomous vehicles today do **!!!!source!!!!**, and another approach needs to be used. It is chosen to sample the route of the first truck as waypoints and send this to the second TB, which follow the tracks of the first one, by driving through the waypoints. The waypoints will be described according to a global reference frame, where each waypoint includes the following information:

- X and Y position
- Heading of the TB
- Timestamp

The X and Y position describes the placement of the waypoint. The heading corresponds to the heading of the vehicle at the waypoint and is equal to the tangent of the driven curve in the specific point. The heading is defined as the angle between the driving direction and the X-axis in counter clock direction. The timestamp is used in order to determine the time until next waypoint and thereby the second TB should drive the distance on same time as the front one.

As described in section II-E1 two different controllers is tested to reach the waypoints. The first controller is driving directly from one waypoint the to next in a straight line. This requires the waypoints to be sampled with a high enough frequency, which will make the error in the heading when reaching the waypoint acceptable small, as seen in figure 1. The second controller drives between the waypoints in curves, which will allow for lower sampling rate of the waypoints, but it still needs to be high enough in order to resemble the original route from the first TB.

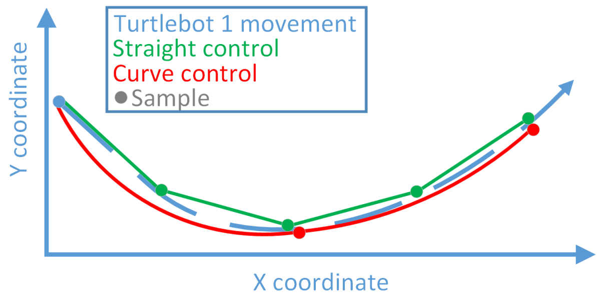

Figure 1. Sampling of first Turtlebots route into waypoints.

B. System structure

The system is based on two TBs, TB1 and TB2, which can be seen in figure 2.

To avoid timing issues, the system have been structured into nodes, such that each node have their own independent timing scheme.

A "remote control handler" node establishes the connection to a PS3 controller and feeds the control inputs to the motor driver of TB1. As the TB uses a differential drive, the inputs are changed to achieve the driving behavior of a truck, see section II-C. The positions of the TBs are measured using Vicon interfaced through MATLAB, and sent to the respective TB at a constant frequency to emulate a GPS.

TB1 sends the waypoints using the "waypoint sender" node, which contains its position, heading and a timestamp. The waypoints is send via WiFi to TB2, where the "waypoint receiver" node listens for incoming waypoints and stores them in a buffer. The buffer is used by the "driving system" node which consists of a "driving controller", a "waypoint updater", a "position receiver" and a "collision avoidance". The "driving controller" is designed to make TB2 reach the waypoints based on the position received from the "position receiver". The "waypoint updater" updates the waypoints used by the "driving controller" whenever a waypoint is reached. The "collision avoidance" is implemented to prevent TB2 colliding with TB1 in cases where it stops. This is further described in section II-E.

C. Front vehicle behavior

The movement pattern of the TBs is different from that of a truck. The TB steers using a differential drive, while a truck's steering can be modelled based on the bicycle model.

A differential drive manipulates its angular velocity by having different wheel speeds, while a truck achieves this by changing the angle of its front wheels.

The translatoric and angular velocities of a differential drive is independent of each other, only limited by the constraints of the motor. The bicycle model requires both a specific wheel angle and a translatoric speed to change its angular velocity.

The angular velocity, ω, based on the bicycle model, can be modelled as

$$\omega = \frac{s}{L} \cdot \tan\phi \,, \tag{1}$$

where s is the translatoric speed, L is the length between the front and rear wheel axles and ϕ is the turn angle of the wheels. Equation 1 relates the two inputs of the traditional vehicle s and ϕ, which is set by the driver through the PS3 controller, to the inputs of a differential drive ω and s.

Using this equation to constrain the movement of the TB such that it emulates a truck, eliminates movement that is not obtainable by the bicycle model, such as turning on the spot.

This ensures the trajectory of the TBs resembles that of a truck, and thereby making the proof of concept test setup more comparable to the actual system.

D. Communication

The communication protocol described in this section covers the communication of the waypoints between the two TB. In this project the available physical medium for the communication is Wi-Fi and it is chosen to use TCP, since this has great benefits for a testing setup. It ensures no packet loss if a connection is available. This is chosen despite the increased amount of overhead, and thereby increasing the bandwidth used by the communication. This is however not a problem, since the frequency for this project will be very low compared to what Wi-Fi and TCP normally is used for, like ethernet. Furthermore TCP handles if bit-shifts has happened in the data to ensure the right data is received.

As this project rely on no packet drops and no errors in data, the communication protocol does not need to take this into account. It is decided to send the message as a text message, where each individual item will be separated with \$. Such that it is formed in the following way:

$$\text{[X]\$[Y]\$[Heading]\$[Timestamp]} \tag{2}$$

When the message is sent as a text message, each character will consist of one byte. In order to define the maximum size of the message, the maximum amount of characters for each item has to be determined. The X and Y position will both be in millimeters and the TB will never exceed 10000 mm in the test setup, which corresponds to a maximum of 5 bytes each. The heading is defined in radiance with a precision of 0,01 rad, which corresponds to 4 bytes. The timestamp is in milliseconds and is defined at a maximum of 6 characters, which corresponds to a maximum of 15 minutes, which is deemed more than enough for testing purposes. All together the message then consists of 23 bytes.

In order to figure out the bandwidth used by the protocol the message frequency is needed. This will however not be determined here as different frequencies will be tested in order to find a good compromise between the performance of the controllers and the bandwidth used. The purpose of this is that two different controllers is to be tested, where it is expected that one of them is better at handling lower frequencies. The bandwidth will therefore be proportional to the message size of 23 bytes and message frequency, when not taking TCP and Wi-Fi overheads into consideration, since this is only chosen for test purposes for this project and would most likely not be used on the real case.

E. Driving system

The driving system is controlling TB2 to the wanted waypoint and consists of four subparts; ”Waypoint updater”, ”position updater”, ”collision avoider” and ”driving controller”, as seen in figure 3.

The ”waypoint updater” updates the current waypoint to reach for the ”driving controller, whenever a waypoint is reached. The TB will not reach the waypoint exactly, and thereby the ”waypoint updater” updates the waypoint, when the TB is within a error radius of the waypoint, consistent of 25 mm, which is deemed acceptable. In this way the ”driving controller” always has the right reference waypoint to control against.

The ”position updater” handles the communication with the Vicon system and updates the current position, such that the three other modules are able to use it.

The ”collision avoidance” is ensuring that TB2 does not drive into TB1 if this stops. For this test setup, a distance controller is thereby enough. The ”collision avoidance” is limiting the maximum allowed speed of TB2, which then stops if it gets to close to TB1.

The ”driving controller” is making the TB reach the reference waypoint at the right time. This is done by using the TBs position, heading and speed from the motors, to control towards the waypoint.

1) Driving controller :

It is feasible for TB2 to go between each waypoint, since the similar front vehicle already has been through the path and thereby there should always be a way to obtain the same state for TB2. It is chosen to try two slightly different approaches for controlling the TB2 towards the waypoint. This is in order to compare the two approaches and thereby conclude which is best in what cases. One of the approaches will use the kinematic equations and assumes an ideal kinematic system. In reality the TBs are dynamic, but the internal controllers have a risetime of XXXX ms, and thereby a kinematic approach is tried. The other approach will handle take the dynamics of the system into account, since it uses controllers able to handle the dynamics.

The first approach calculates a curve between the current state of the Turtlebot and the waypoint. By this it is assumed that the movement to the waypoint can be done in one curve with unchanging curvature. In this way the waypoint should be reached with the desired heading.

The second approach assumes that the difference between the wanted heading in the waypoint and the heading of the TB is small enough to be acceptable. The TB is thereby controlled in a straight line towards the waypoint.

2) Curve controller:

In order to calculate the curve to the next waypoint the differences between the current state of the TB and the waypoint that is to be reached is calculated. This as the following:

$$\vec{p}_e = \vec{p}_w - \vec{p}_c \tag{3}$$

$$\theta_e = \theta_w - \theta_c \tag{4}$$

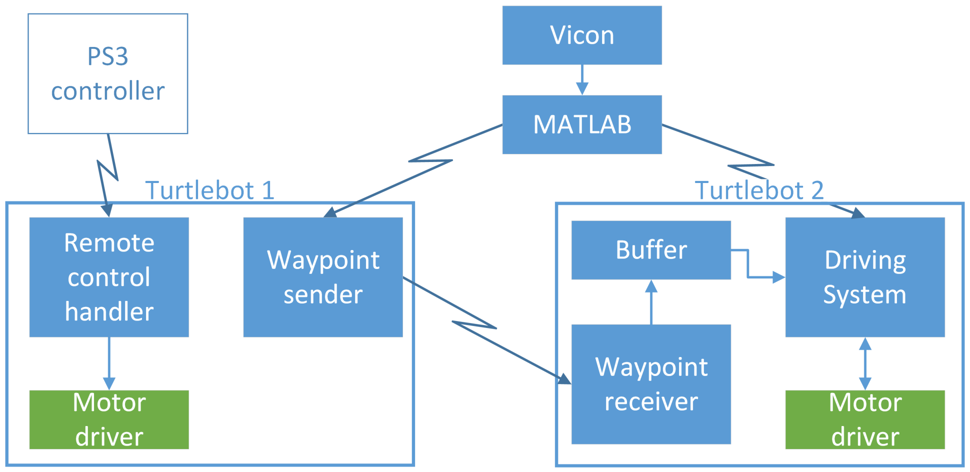

Figure 2. Overall system structure

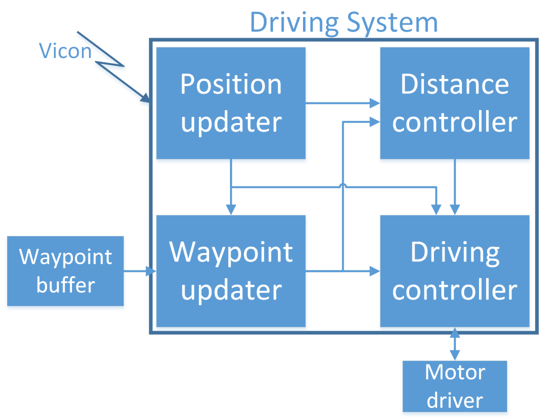

Figure 3. Driving system expanded

$$T_{\mathrm{dif}} = T_w - T_c \tag{5}$$

where p is the position vector $\vec{p} = [x\ y]^{\mathrm{T}}$, θ is the heading and T is time. The subscripts 'e' corresponds to the error, 'w' corresponds to the waypoint, 'c' corresponds the current state of the TB and 'dif' corresponds to the difference.
The angular velocity, ω, needed to reach the waypoint is

$$\omega = \frac{\theta_{\mathrm{e}}}{T_{\mathrm{dif}}} \tag{6}$$

The direct distance, d, between the TB and the waypoint is then calculated as the norm of $\vec{p}_{\mathrm{e}}$, and is used in order to find the radius of the circle to drive on.

$$d = \|\vec{p}_{\mathrm{e}}\| \tag{7}$$

The radius, r is as seen in figure 4, where the cosine relation is used such that

$$r = \frac{d}{\sqrt{2(1-\cos(\theta_{\mathrm{e}})}}\ . \tag{8}$$

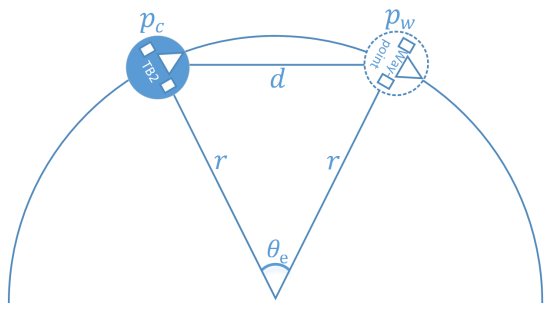

Figure 4. Curve movement to waypoint

The angular velocity and the radius is then used to calculate the translatoric velocity, v, if the angle error is not zero. However if the error is 0, then the TB should just go straight toward the next waypoint according to distance and the time difference.

$$v = \begin{cases} |\omega| r & \text{if } \theta_{\mathrm{e}} \neq 0 \\ \frac{d}{T_{\mathrm{dif}}} & \text{if } \theta_{\mathrm{e}} = 0 \end{cases} \tag{9}$$

This should give the right curve and reach the waypoint as intended under the assumption that the TB drives kinematic according to the equations. This is however not the case as some errors and dynamics will be there and because the sensors will contain noise. Therefore the control outputs will be recalculated multiple times in a feedback loop as seen in the block diagram in figure 5.

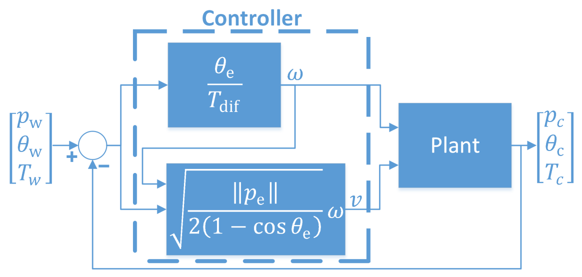

Figure 5. Curve controller

Since TB2 should be limited to a truck behavior like TB1, the angular velocity from the controller is truncated according to equation 1 as the first TB.

3) Straight line controller:
This controller controls TB2 straight against the next waypoint. In order to do this, the references for the controllers needs to be calculated first. The needed heading, θ_{r}, in order to drive from the current position of TB2 to the waypoint is calculated using arctangent.

$$\theta_r = \arctan\left(\frac{y_e}{x_e}\right) \tag{10}$$

The translatoric velocity reference, v_{r}, calculated such that TB2 reach the waypoint at the right time. This is done according to the distance to the waypoint and the time it should take to get there.

$$v_r = \frac{d}{T_{dif}} \tag{11}$$

The TB with its in build controllers can be approximated as a first order system as seen on figure 6.

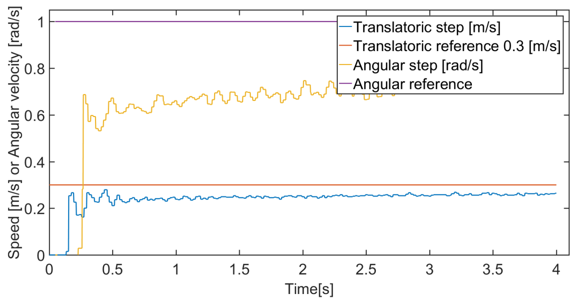

Figure 6. Step response of Turtlebot with in build controllers.

Notes for figure 6: Sorry for hand sketch. The graph should show the measured step response and a comparison with the step response of the approximated first order system. There are 2 systems, one for the translatoric velocity, v, and one for the rotational velocity, ω.

The response is approximated with a first order transfer function, one for the translatoric velocity, G_t, and one for the rotational velocity, G_r. The approximated first order transfer functions is:

$$G_t = \frac{k}{\tau s + 1} \tag{12}$$

$$G_r = \frac{k}{\tau s + 1} \tag{13}$$

In order to obtain the wanted heading and velocity for TB2, PID controllers are used as seen in figure 7. The transfer functions for the system is used to tune the PID controllers to obtain the wanted behavior.

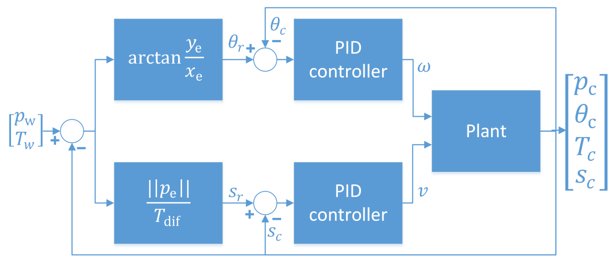
$\begin{bmatrix} p_w \\ T_w \end{bmatrix}$
+
−
$\arctan \frac{y_e}{x_e}$
θ_r
+
θ_c
−
PID controller
ω
Plant
$\frac{||p_e||}{T_{dif}}$
s_r
+
−
s_c
PID controller
v
$\begin{bmatrix} p_c \\ \theta_c \\ T_c \\ s_c \end{bmatrix}$

Figure 7. Block diagram for straight line controller

The references, θ_r and v_r, are updated on every run-through of the controller, since the wanted heading and velocity is different in every run-through, since the system is dynamic and will not react exactly as predicted. Since TB2 should be limited to a truck behavior like TB1, the angular velocity from the controller is truncated according to equation 1 as the first TB. Without this limitation, TB2 would turn as fast as possible against the right heading, however, this can be limited by the driving behavior of the real truck, giving TB2 a not totally straight line approach in the first couple of run-through towards a new waypoint.

F. Test description

The setup will be tested by driving TB1 and making TB2 try to follow in the same tracks. This is done by having a predefined route that the TB1, such that results are comparable. In order to figure out how precise the tracks of TB1 has been followed, the mean square error, MSE, is found as a number of precision. This is test is done several times in order to measure whether the precision changes for each run.

III. Results

The results are to presented here, with graphs of how well TB2 follows in the tracks of TB1.

Thoughts on presentation of results:
A graph like figure 1 where the original route and the driven route by TB2 can be seen, for different sampling rates in order to compare them. This will be shown for each controller. For each sampling rate with each controller the MSE is also found in order to compare them.

It is expected that the straight line controller will be best in, as it will be better to handle the dynamics of the system. The curve control might however be the best with lower message frequencies, as the straight line controller will differ more from the route when doing turns.

IV. Conclusion

It is seen that the straight controller is best in most cases even though the curve controller should be best at handling turns. This is because TB2 is limited the same way as TB1, which means it cannot do too sharp turns, which means that the straight controller does not move in as straight lines as supposed. This can therefore more similar to the actual route.

References

[1] Roozbeh Kianfar, Mohammad Ali, Paolo Falcone and Jonas Fredriksson *Combined Longitudinal and Lateral Control Design for String Stable Vehicle Platooning within a Designated Lane* Department of Signals and Systems, Chalmers University of Technology, SE-412 96 Goteborg, Sweden