Automated Guided Vehicle

Evan Hardester
ECE-6930
USU

Nathan Tipton
ECE-5930
USU

***Abstract*—Project report for Automated Guided Vehicle final project.**

I. Design Overview

The system as a whole consists of the Hybrid Controller for the Automated Guided Vehicle and a MATLAB function block to calculate the value of ‘d’ (vehicle distance from the center of the road).

The variables used in the Hybrid Controller for the Automated Guided Vehicle follow the guidelines found on page 396 of the textbook in Figure 9.13 [1]. The vehicle starts with an initial position x0 and y0 along with an initial angle θ0. The current coordinate position x and y is stored in local continuous variables ‘x’ and ‘y’. The current angle of the vehicle is stored in another local continuous variable ‘theta’. The max speed is defined as a constant variable ‘v’ and is used as an input to the controller. The distance from the center of the road is carried into the system through a variable of type double named ‘d’ as an input into the controller. The edge of the road, or the point at which when reached the vehicle begins to turn, is stored in a variable of type double named ‘e’ and is used as an input to the controller. A start and stop switch is represented with a Boolean variable named ‘u’ (start = true, stop = false) and is also used as an input to the controller. The current position x and y is also stored in discrete output variables ‘x_out’ and ‘y_out’ of the controller to pass on to the MATLAB function block.

The last three variables found in the system that have not yet been covered are ‘x_dot’, ‘y_dot’, and ‘theta_dot’. These variables are implicitly created by Simulink when we defined the local continuous variables ‘x’, ‘y’, and ‘theta’. These variables represent the time derivatives (rate of change) of those local continuous variables.

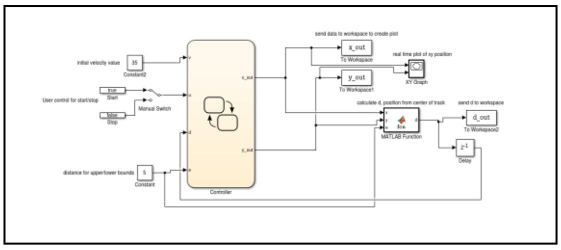


Fig. 1. System overview.

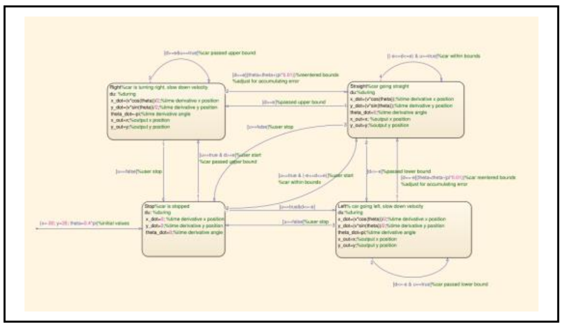


Fig. 2. Hybrid automata of controller

The modeled behavior of the Hybrid Controller for the Automated Guided Vehicle follows the guidelines found on page 396 of the textbook in Figure 9.13 [1] with a few minor changes. The most basic description of the controller is that it contains four states: Stop, Straight, Left, and Right.

Stop: The initial state of the controller is Stop. When the system first enters this state, the local variables ‘x’, ‘y’, and ‘theta’ are set to their initial values and while in that state: x_dot = 0; y_dot = 0; theta_dot = 0. To leave the state Stop, the variable ‘u’ must first be true (start). There are three possible edges that leave the Stop state and all require that the vehicle is set to start (u == true). The first transition is if ‘-e ≤ d ≤ e‘, then it transitions to the Straight state. If ‘d ≤ -e‘, then it transitions to the Left state. If ‘d ≥ e‘ then it transitions to the Right state. We will first consider the case where ‘-e ≤ d ≤ e‘.

Straight: Within the Straight state: x_dot = v cos(theta); y_dot = v sin(theta); theta_dot = 0; x_out = x; y_out = y. There are three possible edges to leave Straight, the first is when u == false which represents stop and will transition to the Stop state. The next two are determined by the current value of ‘d’ and trigger when the vehicle is at or past the edge of the road meaning the vehicle must turn to get back on it. If ‘d ≤ -e’, then a transition to the state Right is taken. If ‘d ≤ e’, then a transition to the state Left is taken. While in state Straight, if ‘-e ≤ d ≤ e‘ then the controller remains in the Straight state. Now consider that case where ‘d ≤ -e’ is true, a transition to the state Left occurs.

Left: In Left: x_dot = (v*cos(theta))/2; y_dot = (v*sin(theta))/2; theta_dot = pi. Within this state Left, there are two possible edges that leave this state, the first is like the first in Straight where u == false which represents stop and will transition to the Stop state. The next is when ‘d ≥ -e’ which means the vehicle is back at the edge or back on the road and enables a transition to the state Straight. If in state Left and ‘d ≤ -e’ remains true, then the controller can remain in this state. Now in state Straight, we already know what occurs here, so now consider the case where ‘d ≥ e’ is true, a right turn must occur so a transition is taken to state Right.

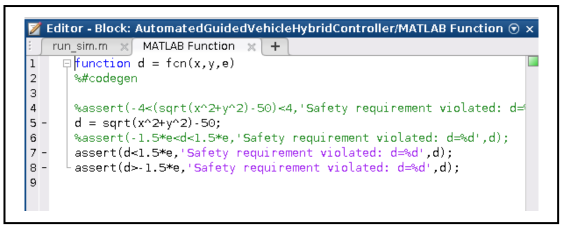
Right: In Right: x_dot = (v*cos(theta))/2; y_dot = (v*sin(theta))/2; theta_dot = -pi. From this state, there are two edges that leave this state, just like state Left. The first is when u == false which represents a stop and a transition to the Stop state will occur. Next is when ‘d ≤ e’ which means the vehicle is back at the edge or back on the road and enables a transition to the state Straight. If in state Right and ‘d ≥ e’ remains true, then the controller can remain in this state.

The main adjustment that was made to the model occurs during the transition back from the Left or Right state to the Straight state. When going from the Left to the Straight state, ‘theta’ is incremented by 0.01*pi. In the case of going from the Right to the Straight state, ‘theta’ is decremented by 0.01*pi. The reason for these adjustments will be addressed later.

It was assumed that the track was a circle centered at the origin with a radius of 50. The initial values for the controller were: v=35, e=5, x0=-30, y0=35, and θ0=0.4π.

II. Enviroment

The model for the hybrid controller’s environment is modeled by a MATLAB function. x_out and y_out are passed to the function from the hybrid controller. Variables x_out and y_out are discrete variables which represent a sampling of the continuous variables x and y inside the hybrid controller. At determined time steps the continuous variables are sampled and the results are output by the controller. These values are used to calculate the vehicles distance from the center of the track, ‘d’. As the vehicle goes straight, the absolute value of ‘d’ increases until the vehicle crosses ‘e’ or ‘-e’. At this point, the vehicle attempts to turn back to the center of the track, causing the absolute value of ‘d’ to decrease. Using the MATLAB function also allowed us to implement the safety requirements into the environment.


Editor - Block: AutomatedGuidedVehicleHybridController/MATLAB Function
run_sim.m
MATLAB Function

```
function d = fcn(x,y,e)
%#codegen

%assert(-4<(sqrt(x^2+y^2)-50)<4,'Safety requirement violated: d=
d = sqrt(x^2+y^2)-50;
%assert(-1.5*e<d<1.5*e,'Safety requirement violated: d=%d',d);
assert(d<1.5*e,'Safety requirement violated: d=%d',d);
assert(d>-1.5*e,'Safety requirement violated: d=%d',d);
```

Fig. 3. MATLAB function modeling environment.

III. Verification

The safety requirement that the vehicle should always stay within a bounded distance, [-1.5e,1.5e], from the track was encoded into the environment using assertions. The vehicle satisfies this requirement at the start of the simulation but eventually fails the safety requirement. The simulation time step was changed in an attempt to pass the requirement. However, this only delayed the failure. The vehicle failed due to accumulated error during the vehicle’s turns. Incorporating the previously mentioned changes to theta accounted for this error and allowed the vehicle to satisfy the safety requirement.

Further testing was done to see how far we could go. The velocity was changed to double the original value. This caused the vehicle to fail the safety requirement. To fix this, theta_dot was increased to +1.5π and -1.5π. This change allowed the vehicle to pass the safety requirement. If the vehicle’s velocity is too large compared to the rate of change of the angular velocity, the safety requirement is failed. The vehicle is not able to turn back towards the track fast enough before passing the safety bounds. By either increasing the rate of change of the angular velocity or by decreasing the velocity we are able to pass the safety requirement.

The safety bounds were then reduced to [-1.1e, 1.1e] and the vehicle was not able to pass the safety requirement. By changing the controller, we were able to pass the safety requirements. The velocity was reduced to half the original value. The variables, x_dot and y_dot, were decreased by a factor of two in order for the vehicle to perform the turns at a faster rate. This causes the vehicle to have sharper turns, reducing the amount of time the vehicle is outside the bounds [-e, e] where the absolute value of ‘d’ is increasing. This allows the vehicle to stay within tighter bounds.

The tightest bound the vehicle was able to successfully stay within, that we tested, was [-1.05e,1.05e]. In order to satisfy these bounds the theta correction value was increased by a factor of 10, theta_dot was increased to -2.5π and 2.5π, and x_dot and y_dot were decreased by a factor of 8 in the left and right turn states.

References

[1] Alur,Rajeev. *Principles of Cyber-physical systems.* MIT Press, 2015