**Design and Simulation of Intelligent Agent for Self-driving Vehicle**

*Tran Thanh Quang, Louis  
National University of Singapore  
Faculty of Engineering  
(Computer Engineering)  
(A0088366N)*

*Tran Minh Hoang  
National University of Singapore  
Faculty of Engineering  
(Electrical Engineer)  
(*A0088452X*)*

*As a requirement for NUS EE4306 – Distributed Autonomous Robotics System (Team 13)*

***Abstract*** *– The capability of detecting obstacles and taking appropriate collision avoidance actions automatically while following the planned path is critical for safe operation of autonomous vehicle. This paper presents the design and the simulation of an autonomous vehicle, which is capable of performing obstacle avoidance actions while navigating using waypoints. Our vehicle must be able to address the toughest challenges in the development of autonomous vehicle which is the automation of overtaking maneuver.*

**I. Introduction**

Autonomous vehicle can play an important role in vehicle active system to reduce traffic accidents. It can avoid some kinds of traffic accidents and improve the road and traffic efficiency at the same time. Lane change and Overtaking Maneuver are the most common behaviors of vehicle. A total of 7,168 accidents[[1]](#footnote-1) were recorded by Singapore Police Force in 2012. As a consequence, 168 were died while other 9106 were injured[[2]](#footnote-2). Most of these cases happened during land change and overtaking maneuvers. These accidents were caused by failing to leave enough distance, overtaking when there was poor visibility, or by not giving way to an overtaking vehicle. Simulating an autonomous vehicle is the first step in developing driving assistance as well as autonomous driving system to help reducing these mistakes made by human.

To simulate such scenario where the vehicle must react differently due to different situation, we use the Formula One (F1) race, where cars have to go through several laps and overtaking slower cars. The difference with F1 race is that there are cars going in opposite direction.[[3]](#footnote-3) The challenge face the autonomous vehicle is to find out when to overtake. The race consists of the straight and the bend part of the route which can create different scenarios for the overtaking process. When it is in the straight part, the vehicle need to estimate the distance it required before performing land changing and overtaking action. Moreover, it must be able to avoid overtaking near the poor visibility regions (i.e. when it is entering the bend part of the route).

The differential drive robot is then used to simulate our real vehicle. The robot is equipped with Compass for heading control. The Global Positioning System (GPS) is used for navigation using waypoints. Moreover, we use Laser Rangefinder (LRF) to sense the surrounding environments which include other obstacles (i.e. other vehicles).

The simulation is done in Microsoft Robotics Developer Studio through the use of Simple Programming Language (SPL) from Kim Young Joon[[4]](#footnote-4).

**II. Design of Intelligent Agent**

*1. Architecture*

When designing the architecture for our autonomous vehicle, we look at how humans organize the driving task and what operation they perform. These operations are then classified into three levels strategic, tactical and control level[[5]](#footnote-5). The strategic level plans the best route to reach the destination. Next, the tactical level makes complex maneuver decisions like turning, stopping, overtaking, etc. Then, the control level performs basic actions (i.e. steering wheel, pressing pedal or brake) to control the vehicle accordingly.

Inspired by human behavior, our intelligent system consists of 3 levels: Navigation, Control and Locomotion.

*2. Navigation System*

The navigation system executes path planning algorithm to generate waypoints that the vehicle can follow to reach the destination. With the help of GPS, the vehicle will perform *“Waypoints Tracking”* operation to find out when I reach a waypoint. The information from this waypoints-tracking system tells the agents when the vehicle is entering a bend or a straight part of the route, when it is entering the poor vision regions (i.e. entering a bend part of the route). Hence, it can make strategic action to *switch control mode*, to alter vehicle speed and to avoid undertaking in the poor vision regions.

***Mode Switching***

**Control System**

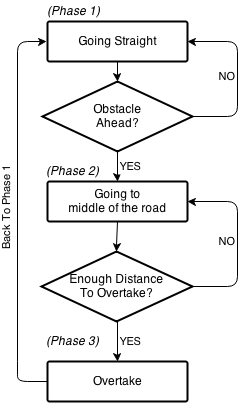
**Locomotion**

***Waypoints Tracking***

**Navigation System**

*3. Control System*

The control system consists of two main driving modes: bend and straight mode. For safety control, overtaking maneurver is only executed on the straight part of the road. The bend mode is implemented using a multi-P controller to maintain the lateral distance and maintain the vehicle heading parallely to the roadside at the sametime. The multi-P controller will be discussed later in the Algorithm session.

There are three phases in our straight mode. The first phase controls the vehicle parallelly to the roadside until it detect another vehicle in front of it. The second phase is then activated to check for overtaking condition. Firstly, the agent will move to the middle of the road to search for moving vehicles around. The distance and velocity of those vehicles are then used to calculate when it is safe to start overtaking. Moreover, the velocity and distance to from the scan are used to help the agent maintain a safe the distance with the vehicle in-front. The third phase performs overtaking maneuver. Those include two land-changing actions and one overtaking action (i.e. going straight at maximum speed).

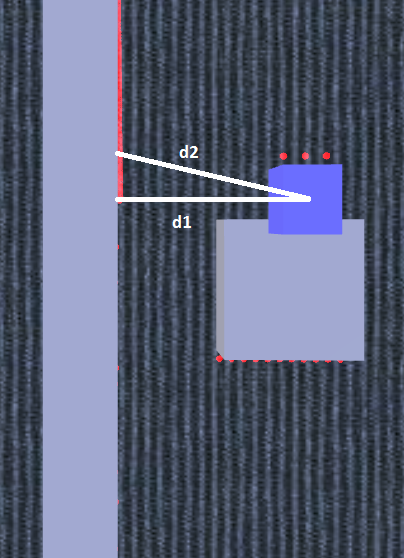
*4. Locomotion*

This system is used for the low-level control of the vehicle. The control of the differential drive robot is done by varying the speed of its left and the right wheel.

**III. Algorithm**

**1. Sensor Data Interpretation**

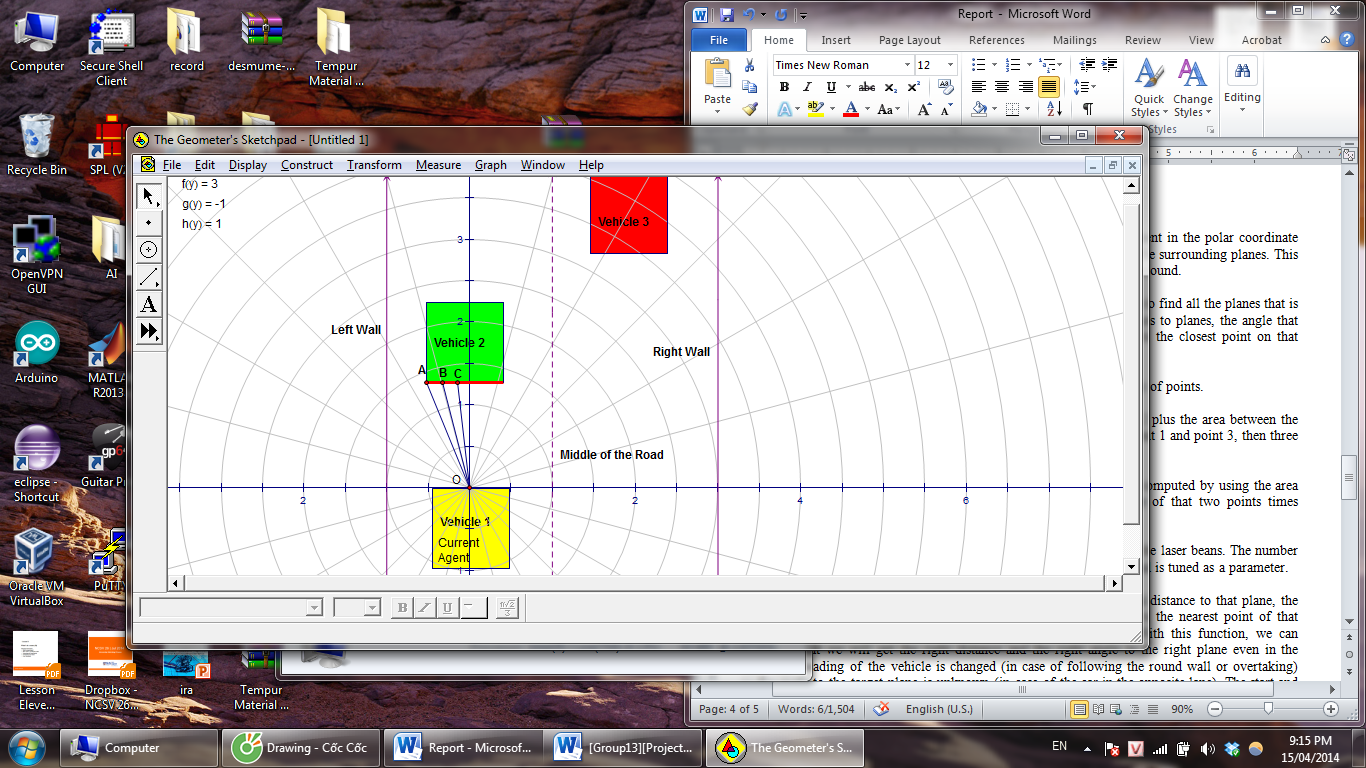
The laser rangefinder (LRF) only returns a set of distances, which is not sufficiently informative for the control of the vehicle. Hence, we designed four functions, getDistanceToWall, getAngleToWall, scanForSpeed and scanForPlane.

1. **getDistanceToWall and getAngleToWall**

In our simulation, two walls are used as roadsides for the vehicle to navigate. Two functions (getDistanceToWall and getAngleToWall) are designed to get the lateral distance as well as the difference in heading between the vehicle and the wall.

The angle and distance to wall are calculated using two fixed laser beams (d1 and d2). Because the two laser line and the wall form a triangular which two sides and the angle between them is known, other data about the triangular can be easily solved.

When we need to calculate the angle to the left wall, d1 is chosen to be the left most laser beam and d2 is a beam that is 15 degree off from d1. Then the angle and the lateral distance to the left wall is then calculated by



1. **scanForPlane**

The distance arrays from the LRF map the points in the environment in the polar coordinate format. By analyzing those points, we can get the information about the surrounding planes. This information is useful later when we need to look for moving vehicles around.

ScanForPlane is a critical function which analyzes all laser beams to find all the planes that is in the range of LRF. This algorithm is implemented based on the collinear characteristic of points in space. Given three points A, B and C, if Area(OAB) + Area(OBC) = Area(OAC), then A, B, C are collinear.

In polar coordinate, Area formed by any two points (for instance A, B) can be easily computed using:

Function scanForPlane finds all the planes by going through all the laser beans. The number of point needed to claim that is a plane depends on the environment and is tuned as a parameter.

The outcome of this function is a set of (1) orthogonal distances to planes, the (2) angle that those planes make with the current vehicle heading, (3) distances to the closest point on that plane and also the (4)starting, ending point of a plane

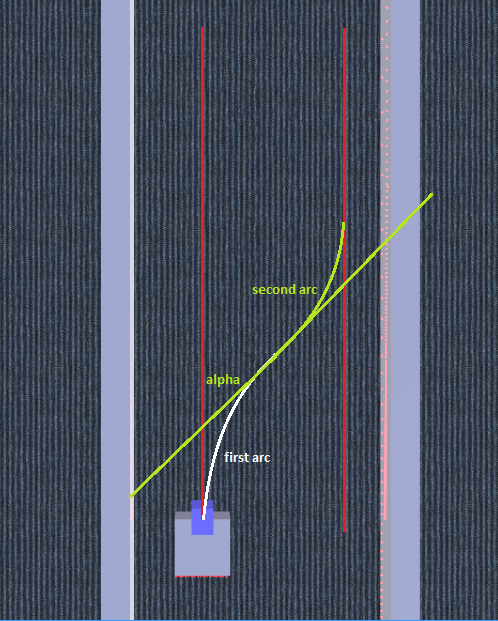
1. **scanForSpeed**

In an overtaking action, it is very important to know the speed of other vehicle on the road. A wrong speed calculation can lead incorrect behaviors, miss of overtaking opportunity or even crashes. This important mission is undertaken by function scanForSpeed.

This function calls scanForPlane two consecutive times to get the orthogonal distance to the planes. Since we know the time difference between two scans, the velocity of those moving vehicles can be easily computed.

**2. Control System**

**A.** **smoothGo**

Any vehicles with more than one non steerable wheel will not be able to move directly in the direction perpendicular to that wheels’ direction so we cannot drive our differential drive vehicle to the side directly. Stop, rotate in place then go then stop and rotate in place again is a possible solution that requires no additional function, but stop to overtake is not acceptable for any high speed vehicle. Because of this constraint, we come up with our own solution, a function called smoothGo that will allow the vehicle effectively move sideways without the need for stop by moving in two consecutive arcs that are tangent with the direction of the road.

The radius of the arcs is calculated to be, with x and z is the forward sand sideway coordinate of the target point in the vehicle coordinate system.

Steering rate is then acquired by the locomotion equation of the differential drive

Where are the speed of left and right wheels; L is the width of the vehicle; R is the radius of the arc.

Compass is used to determine when to change from the first to the second arc and when to finish steering and continue going straight.

When the vehicle reaches the heading the vehicle change it steering rate to the second arc. The value of after arcsine operation is divided by because the arcsine function returns angle in radian while the compass return the heading in scale from 0 to 1.

Finally, the function finishes when the heading of the car reaches the original angle. Compensation is added for the delay of the sensors as well as the car’s movement. The function has given a very good performance with differences value of x and z. However, the performance seems to be very poor with very small z due to the inaccurateness and delay of the compass. A safety checking is used to ensure that the function will not run in case of z is very small and lead to incorrect behaviors.

**B. Proportional controller**

When only the heading of the vehicle needs to be corrected, a simple P controller is used. The controller takes input from function getAngleToWall to make the change in left and right wheel speed of the vehicle.

For example, the current vehicle is going straight at speed v and we would like to correct the heading of the vehicle parallel to the wall.

* The output of the controller is
* Then, the wheel speed is changed to

**C. Multi-Proportional Controller**

When there more than one constrain to satisfy, a single proportional controller is not sufficient to solve the task. Inspired by Fuzzy logic, we come out with a Multi-Proportional controller which combines the output from two P controllers.

* The wheel speed is then

**3. Safety Evaluation**

**A. estimateSmoothGoTime**

Under the smoothGo function operation, movement of the vehicle is complicated so a separate function called estimateSmoothGoTime is needed to compute the total amount of time that the vehicle needs to execute a smoothGo operation. It is computed by the formula:

In the formula above:

* x and z is the forward and sideway distance
* L is the width of the vehicle
* v is the current speed of the vehicle

**B. estimateOvertakingTime**

After computing the presence and speed of other vehicles, our vehicle needs to decide whether it can overtake. A function called estimateOvertakingTime is used to calculate the total time need for the vehicle to overtake another vehicle. Because the overtaking process includes of three periods: go to the right lane, overtake, go back to the left lane, the total overtaking time will be sum of these three periods’ time. Go to the right lane and go back to the left lane action will be carried by function smoothGo, so time needed for these two periods is computed by function estimateSmoothGoTime. In the overtaking period, the vehicle go in a straight line, so time needed will be the distance divided by the speed in that period.

**IV. Result**

The simulated vehicle has been able to follow an arbitrarily bended road while staying stable in its lane, follow a set of waypoints, overtake other vehicles in straight roads, avoid overtaking when entering a bend road and reduce its speed to follow other vehicle when overtaking is not allowed.

Some of the challenges that we faced include the low update rate of the compass which leads to the unstable performance of the smoothGo function when the vehicle is running at high speed. Moreover, view blockage from the in-front vehicle may affect the discovery of the vehicle from opposite land. It may result in wrong overtaking decision made by the agent.

***Challenge in the real world?***

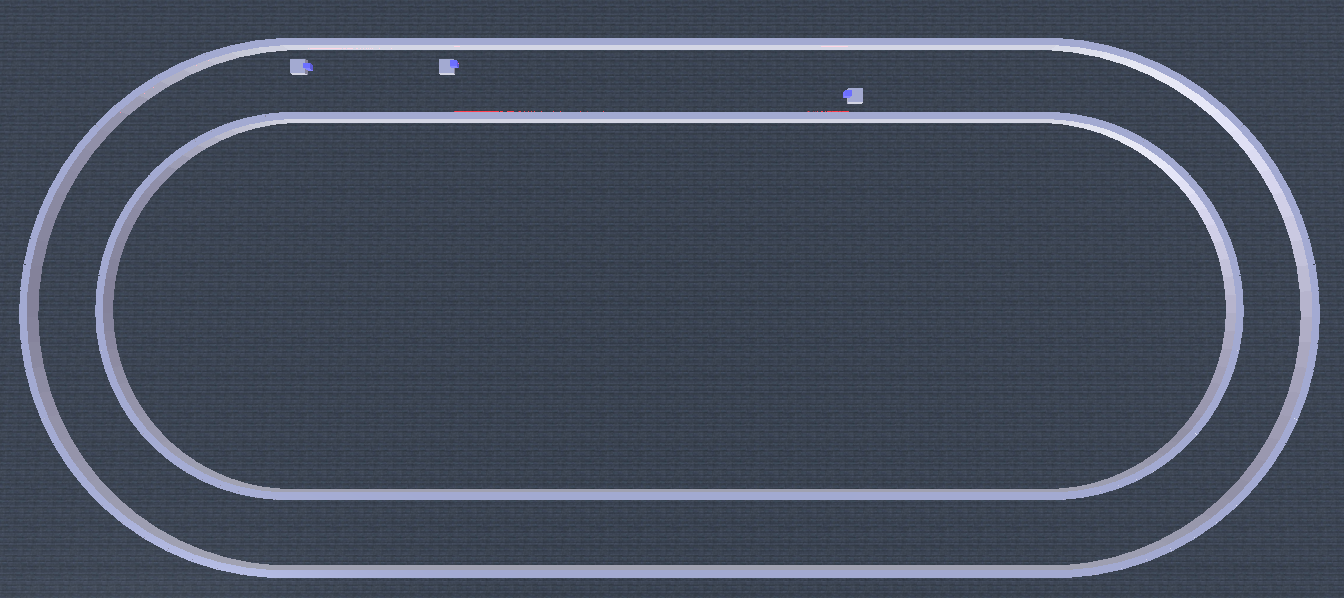
Many idealization has been made in the simulation, they includes the assumption that the left and right wheel give the same power, GPS give the accurate position, compass is accurate (i.e. no drifting), laser rangefinder can detect object from very far away (i.e. more than 10 meters)

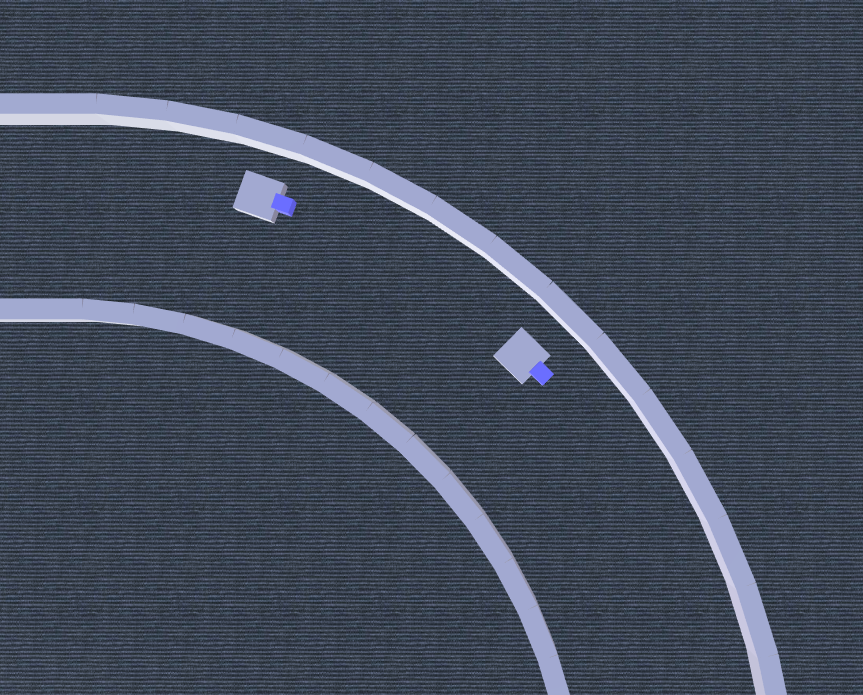
In the real world, there is no wall to refer to and vision processing is required to find out the angle to the roadside as well as distance to wall. The back of the real vehicle is not a plane. Thus, a lot of improvement for obstacle detection algorithm is required.

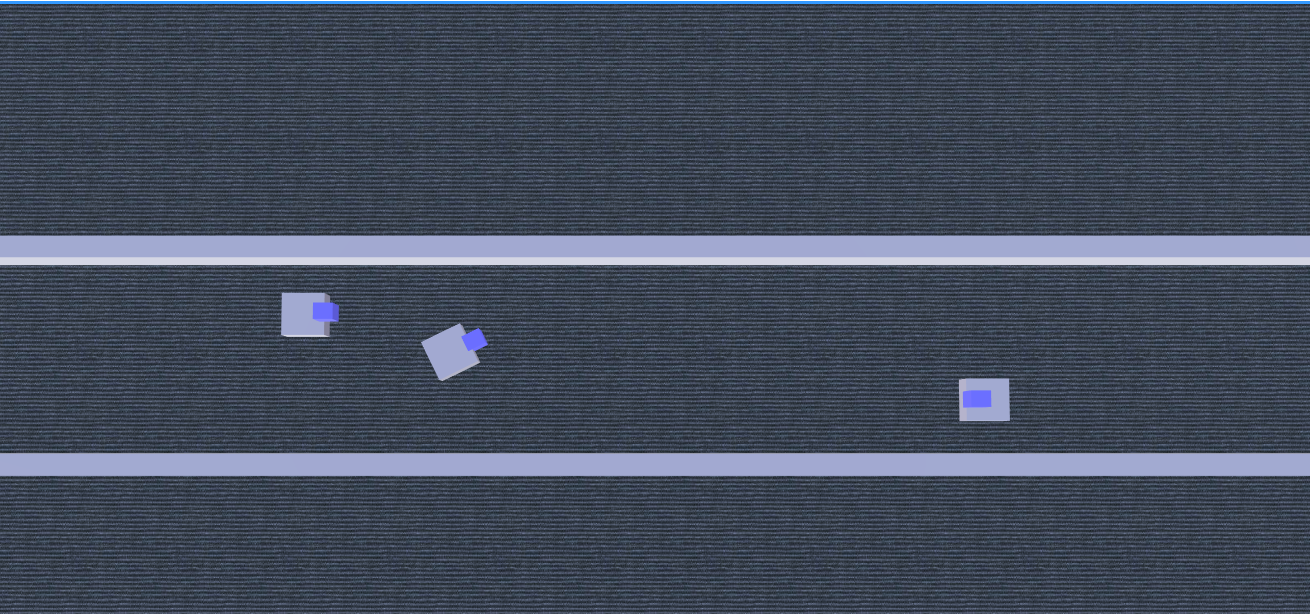
**V. Conclusion**

In this paper, we have presented an automatic driving system that is capable of navigating through waypoints, driving on straight roads and overtaking maneuvers when a slower vehicle appears. A lot of improvement is required when deploying this intelligent agent to real world.

**VI. Screenshot from the simulation**

Formula One (F1) race map

Following slower vehicle when entering a bend road

  
Overtake slower vehicle when vehicle from direction is far away

1. http://driving-in-singapore.spf.gov.sg/services/driving\_in\_singapore/services/statistics.html [↑](#footnote-ref-1)
2. http://www.spf.gov.sg/prints/tp\_annual/2012/doc/12spfa\_casualties2.pdf [↑](#footnote-ref-2)
3. Screenshot of the map can be found at session VI [↑](#footnote-ref-3)
4. HelloApps, http://www.helloapps.com/ [↑](#footnote-ref-4)
5. J. Michon, “A critical view of driver behavior models: What do we know, what should we do?” in Human Behavior and Traffic Safety,L.Evans and R. Schwing, Eds. New York: Plenum, 1985. [↑](#footnote-ref-5)