Designing a driverless car
Gowtham Ashok

Designing a driverless car is a complex task. A car is a big machine that has the potential to cause injury or even death. It has a lot of variables to consider while making a decision. It has to take into account:

Road conditions

- Route
- Obstacles
- Other cars
- Small objects like pedestrians

Law

- Speed Limit
- Collision implications

Fuel

- Cost of fuelling

Destination route

- Distance
- Average speed
- Traffic
- Road conditions for calculated route

It may also have to do some on-the-spot cost analysis as there may be scenarios where there is no best solution. For example, if the car can either hit an animal or swerve off and hit another car, it has to decide which option is the most optimal. This can be a difficult possibility to program. Oftentimes, it may cause unintended consequences which are hard to foresee even by human drivers.

Requirements:

The agent should be able to

- Drive close to or under the speed limit
- Avoid collision with other cars
- Avoid collision with smaller objects like people, animals, road debris
- Should calculate route based on time to destination, tolls
- Should be able to refuel/recharge itself
- Should be able to self-diagnose any problems (mechanical or electrical) and then be able to go to a nearby service center for repairs
- Should be able to recognize road signs (like road work signs) and adapt accordingly

Design:

Assuming the car has all the information it needs to do, a.k.a sensor data, ability to control vehicle motion, map data.
For the sake of simplicity, let's assume that it has the following items:

- LiDAR (to calculate other car positions)
- GPS route calculation (something like google maps)
- Camera mounted on it to detect small objects, read road signs, read red lights
- OBD reader (to read engine speed, RPM, manifold pressure, etc)

I'm going to use "**common sense reasoning**" methodology , to make the agent. This will help the agent make natural inferences about the world, just like we do. It gives a formal structure to the agent, to interpret the world around it, that is, the road system
It has composed of primitive actions, which are like a set of indivisible basic rules. Each primitive action has a frame corresponding to it. For example, accelerating the car is a primitive action. The agent (car) itself is the frame in this case.
An implied action can be a set of primitive actions. For example, swerve left to avoid cow, would be, detect cow, turn on left indicator, change lane to move to left lane, turn off left indicator.

I chose this, since I have come to see that the best drivers are the ones that usually have more experience in driving cars, in varied terrains and countries. It helps in addressing the difficulty of the problem as the commonsense approach would allow us to come to a workable approach, even in situations which are not exactly the same as programmed. We cannot program the agent for all the scenarios, as the number of scenarios to program are too great. Furthermore, if we write a very complex program, it would increase the reaction time, which in itself is dangerous to the agent.

Let's take an example of speed limits. The same speed limit has different meanings in different conditions. In ideal conditions, the posted speed limit may be the best choice to follow. But if there is snow or black ice on the road, it would be better to reduce the speed limit, to allow for more traction on the road. So, we see that in this case, context plays an important role. This is similar to the "eat" example we saw in the lecture.

Let us now design the system, based on the requirements we listed before.

- Drive close to or under the speed limit

  Use camera mounted on car to read posted speed limit.
  Check current speed of agent.
  Check current speed of surrounding cars.
  Verify state/country law for maximum variation allowed. (Many states allow you to go 10mph over the speed limit)
  Check road conditions.

  Every 100ms - 1s, let's use the data collected in these procedures to adjust action.

For example, if agent is going below the speed limit, cars surrounding it are going at the speed limit and the road conditions are ideal, then initiate “accelerate” action.

- Avoid collision with other cars

  Check current speed.
  Check other car speed.
  Check other car stopping distance in commonsense database (Trucks have a big stopping distance, for example)

  Use this data, and infer from accident databases, to predict the probability of accidents based on current conditions, and then initiate any set of actions if necessary. For example, slow down and change lane if other car drives erratically (possibility of drunk driver).

- Avoid collision with smaller objects like people, animals, road debris

  Check current speed and current position relative to object.
  Check object direction and speed.

  Use this data and predict collision. If probability of collision> experimentally_determined_amount then initiate actions to avoid it.

- Should calculate route based on time to destination, tolls

  Calculate shortest route distance.
  Check road conditions from common-sense database (a shorter route is not always the best route. It may have many potholes or may be an accident-prone zone)

  Then the agent can select the best route based on the available data.

- Should be able to refuel/recharge itself

  This would involve a simple if-else statement.
  If amount_of_fuel < fuel_required_to_reach_best_fuelling_station + slack_amount
  Then refuel

- Should be able to self-diagnose any problems (mechanical or electrical) and then be able to go to a nearby service center for repairs

  It should be able to perform a self-inspection based on the commonsense knowledge of the agent’s driving. For example, if it drives through a very rough road with a lot of potholes, then it should self-diagnose or go to a nearby station to check for its suspension condition

- Should be able to recognize road signs (like road work signs) and adapt accordingly

The agent should be able to recognize the different road signs and adjust its speed accordingly. This is similar to different road conditions.

Some of the primitive actions may be:

- Accelerate agent
- Decelerate agent
- Blow horn in agent
- Switch on indicator (left/right)
- Self-clean camera (incase of rain)
- Control Headlights

Some of the implied actions may be:

- Swerve left
- Swerve right
- Refuel
- Go to destination
- Drive safely

In order to populate the common-sense database, we can use existing databases as well as create new databases.

Existing databases:

- Accident reports
- Recording actions of expert drivers and then making valid inferences on their actions
- Traffic law of the jurisdiction they are travelling in
- Footage of car races (legal races such as autocross races and illegal street racing)
- Recording user actions in open-world car games (in games like GTA, Midtown Madness)

New databases:

We can create a new database and perhaps share it collaboratively

- Maintaining an action log, and analyzing traffic tickets based on timestamp.
- Recording vehicle collision data

Therefore, to create a driverless car, we can create a commonsense database and then perform any required actions to drive safely. I came up with a set of requirements that would be required for the car, why the problem is difficult, how commonsense reasoning is an effective solution. Then, I detailed commonsense reasoning in a simple way, provided reasons to use it, and explored how to use it to satisfy the requirements. I then listed some of the primitive and implied actions, and suggested ways to create a common-sense database.