**Self Driving Car Simulator**

**Authors**: Lioz Elmalem 205804693 & Snir Shaharabani 204040687

**Abstract**

A self-driving car, also known as an autonomous vehicle (AV), connected and autonomous vehicle (CAV), driverless car, robo-car, or robotic car, is a [vehicle](https://en.wikipedia.org/wiki/Vehicular_automation) that is capable of sensing its environment and moving safely with little or no [human input](https://en.wikipedia.org/wiki/User_interface).

Self-driving cars combine a variety of sensors to perceive their surroundings, such as [radar](https://en.wikipedia.org/wiki/Radar), [lidar](https://en.wikipedia.org/wiki/Lidar), [sonar](https://en.wikipedia.org/wiki/Sonar), [GPS](https://en.wikipedia.org/wiki/GPS), [odometry](https://en.wikipedia.org/wiki/Odometry) and [inertial measurement units](https://en.wikipedia.org/wiki/Inertial_measurement_unit). Advanced [control systems](https://en.wikipedia.org/wiki/Control_system) interpret [sensory information](https://en.wikipedia.org/wiki/Sensory_information) to identify appropriate navigation paths, as well as obstacles and relevant [signage](https://en.wikipedia.org/wiki/Road_signs).

Why do we need autonomous cars?

Numbers made available by the US government state that 94% of the vehicle accidents are due to human failures.

Road accidents are caused due to a triangle of factors:

the human factor

The person's condition and behavior on the road greatly affect the formation of many road accidents such as: inattention, distraction (for example due to using a cell phone to read or write a message, scanning radio channels, or engaging in GPS programming), delinquent driving, disobedience For traffic laws, driving in a state of fatigue, drunkenness or under the influence of drugs, and social pressure among young people - these prolong the time and distance of the driver's reaction and are among the common causes of road accidents.

The infrastructural factor

The condition of the infrastructure greatly affects the driver's behavior. Unmaintained roads, unclear road signs and vague direction signs, unpaved roads, oil slicks on the road, safety hazards and lack of lighting, can cause road accidents. In Israel, there is a mapping of dangerous roads in which a large number of road accidents have occurred, called "red roads."

The mechanical factor

The condition of vehicles can affect accidents quite a bit. A dilapidated mechanical condition of a vehicle may cause an accident, as a condition of mechanical failure in which a wheel detaches from the vehicle, may endanger the vehicle and cause a rollover. Old vehicles, not properly protected from accidents.

For decades, humans have dreamed of making cars that could drive themselves,

so that travel would be less taxing, and the roads safer for everyone. Toward this

goal, we have made strides in motion planning algorithms for autonomous cars,

using a powerful new computing tool, the parallel graphics processing unit (GPU).

In this project we took the open source of a brilliant guy named Matt Bradley (the link below). Matt has created a simulation of an autonomous vehicle traveling on different routes that the user can add, with the vehicle having to avoid static and dynamic obstacles that illustrate the reality on the roads.

**Why this particular theme?**

We had to choose a project theme from many, but the choice was very clear to both of us.

Simulation, a mathematical modeling of the physical process in real world, plays a

crucial role in almost every field of research. It is also very important for autonomous

on-road vehicle development. For one thing, the traffic scenarios necessary to test the

system are not easily reproducible in real life. For another, the field tests can be very

dangerous, costly and time-consuming.

The development of autonomous vehicles has attracted increasing attention from both academic institutes and the automotive industry. It is believed that autonomous vehicles sophisticated and reliable enough would redefine mobility.

Today a lot of companies have already published a prototype for an autonomous vehicle. Companies like: Waymo, BMW, Apple and many others.

**Research goal**

Our goal is to decipher the behavior of an autonomous vehicle while understanding the mode of travel. However, one must also understand the environment that surrounds the vehicle like pedestrians, bicycles, vehicles, square and so on, to simulate a reality on the road.

This is to answer the question which is better behind the wheel, human or computer?

The main goal is to design a system that can simulate interesting travel routes in order to investigate the nature of driving based on the calculation of a given route that is analyzed along its entire length (detailed below).

**How is works**

This motion planner is mainly based on two Ph.D. theses on parallel lattice-based trajectory planning: *Parallel Algorithms for Real-time Motion Planning* by Matthew McNaughton and *State Lattice-based Motion Planning for Autonomous On-Road Driving* by Shuiying Wang.

**GPU**

A graphics processing unit (GPU) is a specialized electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display device. GPUs are used in embedded systems, mobile phones, personal computers, workstations, and game consoles. Modern GPUs are very efficient at manipulating computer graphics and image processing. Their highly parallel structure makes them more efficient than general-purpose central processing units (CPUs) for algorithms that process large blocks of data in parallel. In a personal computer, a GPU can be present on a video card or embedded on the motherboard. In certain CPUs, they are embedded on the CPU die.

**Motion planning**

Motion planning (also known as the navigation problem or the piano mover's problem) is a term used in robotics for the process of breaking down a desired movement task into discrete motions that satisfy movement constraints and possibly optimize some aspect of the movement. Motion planning has several robotics applications, such as autonomy, automation, and robot design in CAD software, as well as applications in other fields, such as animating digital characters, video game artificial intelligence, architectural design, robotic surgery, and the study of biological molecules

**SL Coordinates and the Lattice**

The simulation and the vehicle run in a standard XY Cartesian coordinate system. However, for certain components, it is easier to work with a coordinate system that is influenced by the shape of the road. Where appropriate, the planner instead uses a station-latitude (SL) coordinate system. The station is the longitudinal distance along the road from some initial point, and the latitude is the lateral offset from the centerline (with negative latitudes to the left and positive latitudes to the right).

Using the SL coordinate system, a lattice is built with equally-spaced points for some spatial horizon distance along the road from the vehicle's current position. Each lattice point has an SL position, a heading, and a curvature, all fully defined by the shape of the road. The lattice points and edges joining them form a graph of trajectories; the motion planner returns the best path through this graph as decided by a cost function made up of various terms.

**Polynomial Spiral Paths**

To allow for smooth driving, any path return by the motion planner must guarantee continuous curvature (meaning no discontinuous rotations of the steering wheel); it's obvious that straight edges cannot be used to connect the lattice points between stations. A polynomial spiral is a curve whose curvature is a polynomial function of it arc-length; it provides the planner with continuous curvature paths, and its coefficients can be quickly estimated using gradient descent. Edges between lattice points are created following some connectivity pattern, and the polynomial spiral coefficients for each edge are optimized in parallel using a WebGL shader. Polynomial spiral edges are also created to connect the vehicle's current position to points on the lattice.

**Obstacles**

Both static and dynamic obstacles are avoided by the motion planner. Static obstacles are considered motionless during the simulation, and dynamic obstacles have their future positions calculated using their velocities and current positions. Obstacles are drawn to an obstacle cost grid with WebGL, with the dynamic obstacle cost grid having multiple "frames" representing different ranges in time. To ensure safe movement around obstacles, their sizes on the grid are dilated into two zones: the smaller collision zone and the larger hazard zone. Paths traveling through the collision zone have infinite cost and are pruned during the graph search. Paths traveling through the hazard zone have an increased cost, but are still feasible. Additionally, the dilation size is larger along the longitudinal direction compared to the lateral direction. This allows the vehicle to have a smaller latitude distance from an obstacle but a higher station distance (i.e., it is safe to be only a few feet away from a vehicle in an adjacent lane on a highway, but it is not safe to be a few feet behind a vehicle on a highway).

**Graph Search**

Each 2D station-latitude lattice point is augmented into a graph vertex with three additional dimensions: a time range, a velocity range, and a constant acceleration profile. Since the search is conducted in a dynamic environment, time and velocity dimensions are added to the graph vertices. However, to avoid an exponential blowup in the size of the search space, the estimated time and velocity extrema of the planning horizon are discretized into only a few ranges. Out of all the incoming edges terminating at a specific time range and velocity range of a vertex, only the best trajectory is kept. Even though this can potentially discard optimal paths through the graph, edges falling into the same range are effectively equivalent so that the discovered path should be only slightly suboptimal.

Each graph vertex also has a corresponding acceleration profile dimension that determines the change in velocity that happens over the length of incoming trajectories. Several acceleration profiles are used, such as constant hard and soft acceleration/braking and computed accelerations to reach a target velocity like the configured speed limit or a complete stop.

**Cost Function**

Each trajectory edge that connects two graph vertices has an associated traversal cost. This cost is calculated by sampling the polynomial spiral path into distinct points (say every 0.5 meters) and summing the cost terms of each individual point. Various cost terms are used, penalizing things like: proximity to static or dynamic obstacles, lateral offsets from the center of the lane, driving in the non-preferred lane, exceeding the speed limit, accelerating or braking too hard, high lateral accelerations, etc.

**Dynamic Programming**

The assumption is made that the vehicle will only drive forwards along the road, meaning that its station is monotonically increasing. Based on this assumption, determining the total cost of a path that ends at a vertex at some station requires only the costs at the previous stations to have been computed beforehand (the cost is not dependent on later stations). This property of optimal substructure allows us to use dynamic programming to calculate the best path through the graph. The graph search starts with the vertices at the first station and iterates forwards, using the optimal paths found at the previous stations to extend the solution until the end of the spatial horizon is reached.

**Our extensions**

In Matt Bradley's open source we added another parameter to the car called score, the purpose of which is to calculate the vehicle's travel score. This way we can differentiate between autonomous travel and manual travel.

The score starts at 100 and for each collision / malfunction in the obstacle, decreases by 5 points.

The user can choose between manual driving with the keys: W, S, A, D and drive the route as he wishes or activate autonomous driving.

Another thing we added is the ability to debug the program with a logger. Once in the program, we can log into “developer tools” and watch logs in real time.

The prints are the vehicle parameters, such as: position, speed, angle, score and so.

We also added a diagram that characterizes the project according to its departments. This way you can easily understand the workflow. The diagram called “arkit\_all.svg” in our Github (the link below).

One last thing we added are more varied driving routes that describe problems on the road on a daily basis.

This is another way to differentiate between autonomous travel and manual travel.

**Difficulties while solving**

We tried to add an infinite circular trajectory but we did not succeed because in this simulation there must be a beginning and an end to the trajectory so that the autonomous vehicle will look for the short and correct trajectory to the destination between the obstacles.

We also tried to add a test drive of an autonomous vehicle next to a manual vehicle. Here, too, we found difficulty in realization.

The problems stemmed from the language code comprehension problems and the way the entire program was run. Despite this, we would like to say that we are still trying to complete these additions.

**Conclusions**

Today, the vast majority of road vehicles are used by humans. As noted in the abstract, one of the most significant causes of road accidents are humans.

In this project we discovered that an autonomous car knows how to drive on many and varied tracks with different obstacles that simulate pedestrians, cyclists, cars and other objects.

In most cases the autonomous vehicle knew how to slow down and recalculate a route when necessary, even when we put in a sudden obstacle, like a child running down the road.

Unfortunately, there are an infinity of cases that can happen in reality, when humans, animals and autonomous vehicles are combined.

Our conclusion is that if all the vehicles in the world were autonomous, the amount of accidents caused by the human factor driving the car, would decrease. But the combination of people and autonomous cars on the road will not significantly reduce accidents and deaths as expected from an autonomous vehicle network that will be run on its own.

Maybe if they were all robots, the situation would change ....

**Our GITHUB link:**

* [**https://github.com/LiozElmalem/SDC-Simulator**](https://github.com/LiozElmalem/SDC-Simulator)

**Credits**

* <https://github.com/mattbradley/dash>
* <https://ri.cmu.edu/pub_files/2011/7/mcnaughton-thesis.pdf>
* <https://www.inf.fu-berlin.de/inst/ag-ki/rojas_home/documents/Betreute_Arbeiten/Diss-Shuiying.pdf>
* <https://en.wikipedia.org/wiki/Graphics_processing_unit>
* <https://en.wikipedia.org/wiki/Self-driving_car#Self-driving_car>