Multi-Vehicle Planning using RRT-Connect*

Rahul Kala and Kevin Warwick
School of Cybernetics, School of Systems Engineering,
University of Reading, Whiteknights,
Reading, Berkshire, United Kingdom
rkala001@gmail.com, k.warwick@reading.ac.uk

* Extended version of the paper originally published in CIS2011

Citation: R. Kala, K. Warwick (2012) Multi-Vehicle Planning using RRT-Connect, *Paladyn Journal of Behavioural Robotics*, 2(3): 134-144.

Final Version Available At: http://link.springer.com/article/10.2478%2Fs13230-012-0004-5

Abstract
The problem of planning multiple vehicles deals with the design of an effective algorithm that can cause multiple autonomous vehicles on the road to communicate and generate a collaborative optimal travel plan. Our modelling of the problem considers vehicles to vary greatly in terms of both size and speed, which makes it sub-optimal to have a faster vehicle follow a slower vehicle or for vehicles to drive with predefined speed lanes. It is essential to have a fast planning algorithm whilst still being probabilistically complete. The Rapidly Exploring Random Trees (RRT) algorithm developed and reported on here uses a problem specific coordination axis, a local optimization algorithm, priority based coordination, and a module for deciding travel speeds. Vehicles are assumed to remain in their current relative position laterally on the road unless otherwise instructed. Experimental results presented here show regular driving behaviours, namely vehicle following, overtaking, and complex obstacle avoidance. The ability to showcase complex behaviours in the absence of speed lanes is characteristic of the solution developed.

Keywords: Autonomous vehicles, rapidly exploring random trees, RRT-Connect, multi-robot path planning, coordination, robocars, planning, intelligent vehicles.

1. Introduction

The relatively high number of road accidents coupled with improved technology has led to the possibility for intelligent vehicles to replace manually driven vehicles. Such vehicles either aid the human driver or drive completely without human input [1, 2]. Further developments in autonomous vehicles bring us closer to a traffic scenario wherein all vehicles on the road would be autonomous. In this scenario vehicles can talk to each other over ad-hoc networks via an inter vehicle communication scheme which allows them to collaboratively formulate an optimal travel plan [3]. This is unlike human drivers where driving gestures, horn blows, and other indicators are used for driving, as well as for adherence to traffic laws. This autonomous ability opens up new dimensions from a planning perspective.

At present the major focus of the planning community for intelligent vehicles is to enable a vehicle to decide its speed lane of travel. Algorithms for detecting marked speed lanes or to segment out the road region and guess the speed lanes are widespread [4]. Once a decision to stay in a speed lane or to switch lanes has been made, control algorithms drive the vehicle accordingly. Fundamentally control algorithms for driving within a speed lane scenario attempt to maintain the vehicle's position laterally and longitudinally. Lateral position deals with the maintenance of a speed lane and changing speed lanes, while longitudinal control deals with speed control and vehicle following behaviour [5].

Our modelling of the problem denies the notion of speed lanes. Considering that a significant part of the road network in most economies relies on single lane traffic per side of travel and in such situations lane changes are impossible. The notion is that autonomy may make vehicles vary in speed capabilities and size, depending upon their purpose, precision, control, and autonomy. Speed lanes are constructed assuming a fixed width of vehicle and may well make the road underutilized. Also, constant overtakes are important when the traffic is too diverse. As a result it may be 'painful' for a fast vehicle to follow a slow vehicle. While traffic in most economies is too uniform to display any such characteristics, motivation for this research has been gained from Indian traffic which is highly chaotic [6]. In such circumstances a large number of two-wheeler and three-wheeler vehicles, along with large transport vehicles, give a diverse scenario to traffic in its entirety and hence

speed lanes are not always (sometimes even rarely) followed. Stating the invalidity of the speed lanes makes it essential to consider the lateral movements of the vehicle as non-discrete quantities and this greatly increases the complexity of the planning problem.

The other important aspect of our modelling scenario is the use of a fairly complex obstacle framework. It is relatively easy to avoid a single obstacle on the road [5], however many scattered obstacles on the road give a fairly complex grid of possible paths along which vehicles may travel. Consequently multiple vehicles need to collaboratively decide the manner of going about the situation such that the overall travel plan is optimal. Vehicles, differing by size, may have their own trajectories which are optimal as per their own analysis, given the motion of the other vehicles as per the coordination strategy.

The complex problem formulation shifts the focus of our research towards the employment of algorithms used for robotic path planning, rather than those specifically used in vehicle planning where the availability of speed lanes is assumed. The important aspect here is to consider the dynamic nature of the scenario wherein the vehicles may be driving at high speeds. This means the planning algorithm has to be fast or operate in real time. Hence, reactive and behavioural approaches (e.g. [7, 8]) are widely used for planning in such scenarios. These techniques make the robot or vehicle react on sighting any obstacle or another vehicle and make any necessary adjustments in its position. Potential methods [9] are further used for such scenarios where every obstacle or another vehicle has a repulsive potential which decreases with distance while the goal has an attractive potential. It would be fairly simple to model these algorithms to attract the robot to travel in the longitudinal direction, while the lateral movements are controlled by reacting to obstacles or vehicles repulsions. However a limitation of these algorithms is their completeness. It is not only possible but in some cases very easy for the robot to get struck in a region. Consider a fully autonomous vehicle travelling all by itself on the road. If it gets struck, human aid may not be available to re-align it in the right direction and restart it.

Evolutionary and graph search based techniques [10] are clearly out of the question owing to their high computational costs in high dimensionality maps. Hierarchical algorithms [11, 12] may certainly help in better analysis of the maps and building the solution iteratively to ensure that results can be computed rapidly in small execution times. For real time operation the application of these algorithms would mean using initial solutions or solutions developed on smaller resolution maps. However traffic scenarios are path following approaches rather than goal seeking approaches and reducing resolution would mean having a fairly poor idea of the road width which affects decisions regarding overtaking a vehicle or following it, selecting obstacle avoidance strategies, etc.

Hence the attempt here is to look for a complete and real time algorithm. Some similar research issues are however currently in focus in recent literature for robotics in general. Sampling based techniques are a good alternative that cater to the requirements. Hence we have selected Rapidly Exploring Random Trees (RRT) [13, 14] as the base algorithm for our solution. The algorithm starts with the current position as the source, which is the root of a tree data structure that the algorithm generates. At every iteration the algorithm proceeds by selecting a random point in configuration space and extending the nearest node in the tree towards it by some step size. The algorithm stops on reaching the goal or a point very close to the goal.

Kuwata et al. [15] presented an RRT for solving the problem of planning the trajectory of a vehicle. They used the solution from the MIT entry in the DARPA Urban Challenge 2007 with the vehicle named Talos [16]. The vehicle finished 4th in the race. The algorithm developed enabled a vehicle to show overtaking, lane change, obstacle avoidance, and parking behaviours. The chief difference between Kuwata et al.'s work and our work is the presence of multiple vehicles, which puts emphasis on formulating an effective coordination strategy. Our solution is more suited for unstructured and curved roads, which is the result of a different configuration space representation technique. Further Kuwata et al. converted a mechanism to predict the motion of a vehicle for every control signal which enabled them to plan the low level control signals specific to the vehicle. We develop here a framework in which planning and control are loosely bound. The planning algorithm generates a smooth path which may be fed as input to any control algorithm. A generalized scheme is presented in the works of Chakravorty and Kumar [17] who used RRT and other sampling based approaches. The generalized versions of these algorithms can incorporate vehicle dynamics into the planning to result in plans which are feasible considering vehicle control.

The major contributions of this work are (i) Inspired by the general motion of vehicles in traffic, we have proposed here a planning strategy which is biased towards a vehicle's current lateral position. This enables better tree expansion and connectivity checks in RRT-Connect. (ii) We have also proposed a path cost metric which can be computed with a small execution time and aids in generating non-holonomic solutions with large

separation between vehicles and obstacles. (iii) We have proposed a decision making module for choosing between vehicle following and overtaking behaviours. The module relies on fast planning lookup. (iv) We have proposed a new problem modelling framework for intelligent vehicles consisting of multiple complex obstacles, multiple diverse vehicles, all in the absence of pre-defined speed lanes.

This paper extends our previous work [18] where RRT was used for the planning of multiple vehicles. The key differences between this work and the published work are (i) the procedure to use spline curves from source to the extended node and hence checking the feasibility and non-holonomicity of the (so far prospective) path has been replaced by an approximate algorithm that checks for these validity constraints only at the extended node without the generation of splines. Considering the large number of failed and successful node extensions, this saves a significant amount of time, though the solution generated is approximate. (ii) An additional local optimization algorithm has been added after the approximate path is generated, which checks the validity at every points and hence returns valid exact paths. This replaces the need to run an entire time costly algorithm for number of times for optimality considering that optimality is not intrinsic in RRT functioning. The resultant algorithm is hence fast and better in terms of path length. (iii) RRT is replaced by RRT-Connect which is more suited for trivial vehicle following behaviour for which it takes less, and indeed practically no, computation time. Considering that a significant amount of any journey involves vehicle following, this is a big boost to overall computation time. (iv) The planning algorithm needs to alter the travel speeds of the vehicles in case it is not possible to generate a feasible plan with all vehicles travelling at their preferred speeds. Earlier we proposed an iterative decrease in speed at each iteration which resulted in a computationally heavy algorithm. In this paper we take a more heuristically 'intelligent' approach where we may guess speed depending upon the possibility to avoid, overtake or follow another vehicle.

The paper is organized as follows. In section 2 we define the problem and in section 3 discuss the proposed solution. Section 4 deals with experimental results whilst section 5 discusses some related works. Some discussions on our results are given in section 6 whereas in section 7 we give some the concluding remarks.

2. Problem Formulation and Assumptions

The planning of autonomous vehicles is done at multiple levels. The strategic level deals with deciding the places to visit, order of visit of places, etc. Meanwhile the sub-strategic level deals with deciding the roads to take; the decision on which may depend upon traffic prediction, expected travel speeds and/or road length. The lowest tier of planning deals with the generation of a feasible trajectory for each vehicle. A separate simple or complex control system may be required to physically drive the vehicle. The middle tier planning may deal with path validation, re-planning (whenever required), and iteratively building up the trajectory using the lowest level planning. In this paper we only focus upon the lowest level of planning which is trajectory generation. This planning may be different for merging regions, intersections, parking, blocked road and normal road scenarios, in which we only focus on normal road scenarios.

It is assumed that a segment of the road is available, which is the input to the algorithm. The map may be processed to extract left and right boundaries of the road segment. Any number of obstacles of any size and shape may lie in the road segment. However the obstacle framework should not block the complete road, or be such that no trajectory is possible for any vehicle. The arrival times, speeds, orientations and sizes of vehicles are available as they enter the segment of road being planned. For simplicity we assume that all vehicles are rectangles of size l_i x w_i, non rectangular vehicles may then be bounded to the smallest fitting rectangle. The vehicles are all assumed to be car-like with 4 wheels and a steering control. The purpose of the algorithm is to generate a valid trajectory from the current position to the end of the road segment, such that no two vehicles collide and no vehicle collides with a static obstacle. The generated trajectories need to follow the non-holonomic constraints of the vehicle. While planning the algorithm may only account for vehicles into the road segment and not vehicles that enter the road segment at a later time.

The road segment planning when repeated over the entire road may lead to safe travelling of the vehicle from one point to the other. While planning each segment (part of the entire road), the planning algorithm's vision is restricted to that segment only. Hence it is possible that the planning algorithm generates a trajectory such that the vehicle is placed at a position from which it cannot overcome obstacles when it reaches the end of the planned segment. Consider an obstacle-free road segment with only one vehicle, which may prefer to drive straight. On reaching the segment end (also the start of the next road segment) it may discover an obstacle very close. It would have been better if the vehicle had known about the obstacle earlier. For similar reasons we do not divide the road into disjoint road segments. Rather we divide the road into overlapping road segments. The

vehicle is planned at the start of every road segment. As the segments are overlapping, planning of a segment invalidates the last sections of the trajectories of the previous road segment plan.

It is assumed that an inter-vehicle communication mechanism exists between vehicles within the road segment being planned. This allows for the vehicles to communicate with each other. Communication is however regarded as expensive and hence only limited information may be transferred using this mechanism.

The prime factor used to measure the algorithm's success is its ability to generate valid trajectories for all vehicles, if such trajectories do exist. The algorithm attempts to minimize the total time of travel of the vehicles which means the generation of smoothest and shortest trajectories. The algorithm should further attempt to keep enough separation on all sides of each vehicle.

3. Algorithm

3.1 Coordinate Axis System

The scenario is assumed to be available as a map in which the algorithm works. We redundantly use two coordinate axis systems. The first is the cartesian coordinate system (XY) which is the system in which the map is available as an input. This system is used for trajectory generation, local optimization, and vehicular motion. The other is the road coordinate system (X'Y'). In this system X' axis is taken to be the longitudinal direction of the road. In case the road is curved, this axis will also be curved. For algorithmic purposes, we take the X' axis as the first road boundary. The Y' axis meanwhile is taken to be lateral direction of road. The width of the road may not be constant. Hence we take the Y' axis as the ratio of distance of a point from the X' axis in the lateral direction to the road width at that point. This axis system is used to generate samples in the RRT algorithm. For the algorithm to perform well it is important for samples to be generated within the road segment. For a curved road the feasible region would be a small part of the entire map in the cartesian system. The road coordinate system gives a technique for ensuring that all samples are generated within the road knowing the bounds of X' (0 to road segment size longitudinally) and Y' (0 to 1). The two axis systems are shown in Figure 1. Consider point P(x,y) in the cartesian coordinate system. The corresponding point P(x'y') in the road coordinate system is given by equation (1).

$$P(x, y) = P(x' y') = P\left(x', \frac{a}{w} \right) \tag{1}$$

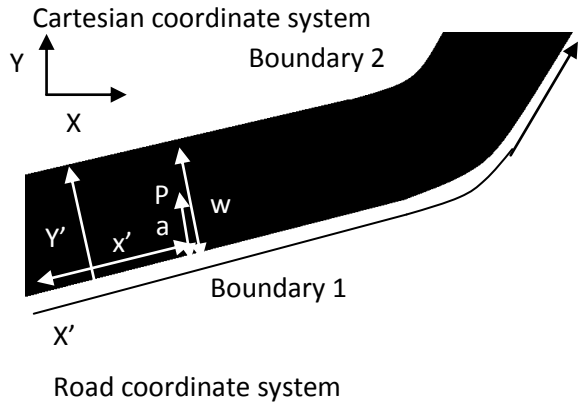

Figure 1: Cartesian and Road Axis System (Source: [18])

Inter-conversion of any point between the axis systems is an important task. Conversion from road to cartesian coordinate systems involves finding corresponding points on the two boundaries and hence computing the ratio. For conversion from road axis to cartesian axis system we find the corresponding lateral point on the X' axis by a small local search and measuring the road width.

3.2 RRT Connect

The basic path is generated using RRT Connect algorithm [13]. The hypothesis is that a vehicle prefers to keep itself intact in the same position on the Y' axis. This means that if the road is obstacle free and there are no other vehicles, the planning algorithm would generate a trajectory parallel to the road (assuming a smooth X' axis), such that the distance of the vehicle from the two boundaries (as a ratio) is always constant. While driving in the presence of speed lanes this corresponds to keeping to one's own speed lane.

The RRT Connect algorithm starts with the initial state or the root of the tree as the source or current position of the vehicle. The vehicle is already facing in given direction and hence the first step it must take is in the same direction. This fixes the first extension of the root which is also the only child of the root. In subsequent iterations we generate random samples using the road coordinate axis system. The tree formulated so far is searched and the closest node to the generated sample (P) is found. A new node (Q) is generated by extending P towards the random sample by a magnitude of *stepsize* which is a parameter of the algorithm. Q is added to the tree with P as the parent if (a) Q is not already in the tree. Nodes very close in the solution space are taken as being the same node. (b) Q ensures that the vehicle can be safely placed at it. This means that a vehicle placed with its centre at Q aligned in the direction of line PQ should be obstacle free and should not collide with any other vehicles. (c) Q ensures kinematic constraints when travelling from a parent of P (say R) to Q via P. The curved structure formed by these three points gives an indication of the maximum speed with which the vehicle may travel in the region considering its kinematic constraints [19]. The maximum speed is given by equation (2).

$$v = \min\left(\sqrt{\frac{\rho}{k}}, vmax\right) \tag{2}$$

in which ρ is called the frictional constant which enables vehicles to make turns, *vmax* is the preferred speed with which the vehicle drives. *k* is the curvature that the set of points (that is R, P and Q) which may approximately be given by equation (3).

$$k = \left\| L_1(d) + L_2(d) - 2P \right\| \tag{3}$$

Here $L_1(d)$ represents a point at a distance of *d* from P along the line L_1 connecting P to R. Similarly $L_2(d)$ represents a point at a distance of *d* from P along the line L_2 connecting P to Q. The value of *d* is chosen so that it is large enough to approximate the prospective curve.

For the node to be regarded as feasible it is important that the vehicle need not reduce its speed. Hence *v* given by equation (2) should be equal to the preferred driving speed *vmax*.

After adding every node to the tree we check if it would be possible to reach the end of the road segment being planned by travelling parallel to the road by keeping the same relative position laterally. In the case when this travel is collision free and obeys the kinematic constraints, the algorithm is terminated. This results in a great saving of time as while travelling in an obstacle-free road segment, or while travelling in vehicle following mode, the algorithm needs to do minimal computations and the path may be returned with the first addition of a node in the RRT.

The algorithm stops when the goal state has been found. In this algorithm the goal is taken to be any point at the road segment end. The algorithm proceeds for a total number of *maxiter* iterations which includes iterations in which the generated node could not be added due to infeasibility.

It may be computationally very expensive to search the entire solution space (which is the entire road segment) in trying to reach the goal from the source. Consider the ideal path that the vehicle would have followed in the absence of obstacles and other vehicles and kinematic constraints as per the hypothesis. This is a line parallel to the X' axis with the same position along the Y' axis. We bias the algorithm to search in the region around this ideal path. In most simple scenarios involving a single obstacle or vehicle it should be possible to generate feasible paths by minor deviations from this path. Hence at every iteration some points are generated and are weighted by their deviation from the ideal path or simply their deviation in the Y' axis to current position of vehicle. One of these points is selected using Roulette Wheel Selection. This ensures biasness towards the ideal

path, at the same time the search process has exploratory potential for scenarios where a valid trajectory may only be possible after large deviations. The RRT-Connect algorithm is given as algorithm 1 and 2.

Algorithm 1: RRT-Connect(source, time, vmax)

tree->root ← source
tree->root->time ← time
rootChild ← point at distance of vehicle's length from root at current angle of orientation
rootChild -> parent ← root
rootChild -> time ← time + length/vmax
if checkConnect(tree, rootChild)
    return path as computed by tree and returned by checkConnect
end if
repeat for a maximum of *maxiter*
    Samples ← Generate random samples in road coordinate system such that probability of selection of sample S = | Y'(S)-Y'(source) |
    Select a sample S by roulette wheel selection
    P ← node nearest to S
    Q← node by extension of P in direction of S
    t_2 ← P->time + stepsize/vmax
    if no point close to Q lies in tree $\wedge$ vehicle placed at Q is obstacle and collision free at time $t_2 \wedge v = v_{max}$
        Q->parent ← P, Q->time ← t_2
        tree ← tree $\cup$ Q
        if checkConnect(tree, rootChild)
            return path as computed by tree and returned by checkConnect
        end if
    end if
end repeat
return null

Algorithm 2: CheckConnect(tree, node)

P_1← node
P_2← point at a distance of stepsize along X' direction from P_1
t_2 ← P_1->time + || P_2-P_1||/vmax
while P_2 is not out of road segment
        if vehicle placed at P_2 is obstacle and collision free at time $t_2 \wedge v = vmax$
            P_2->parent ← P_1, P_2->time ← t_2
            tree ← tree $\cup P_2$
        else return null
        end
        P_1 ← P_2
        P_2← point at a distance of stepsize along X' direction from P_1
        t_2 ← P_1->time + || P_2-P_1||/vmax
end
return path

3.3 Local Optimization

The RRT by design focuses upon rapidly finding a valid path, but it gives no concern to the optimality of the generated path. Hence optimality needs to be added externally in the algorithm. Both global and local optimizations are of interest here. Global optimization outputs the correct strategy for a vehicle to avoid obstacles and other vehicles. It suggests for each obstacle or vehicle encountered whether to avoid it from its left side or right side. Local optimization then deals with computing exact points and magnitudes of turns.

The RRT algorithm outputs a path which is further optimized by a local optimization algorithm. The goal of the optimization algorithm is to minimize the path length while still trying to keep the vehicle separated from other vehicles and obstacles by some distance. The algorithm searches through a number of iterations and stores the best path generated so far. At every iteration the algorithm attempts to deviate from the points represented at that time as the best path (excluding source, source's child and goal) by a factor and evaluates the resultant path cost. If the resultant path has a lower overall total path cost, the algorithm replaces the best path with that newly calculated. The deviation factor is chosen from a Gaussian distribution, the maximum deviation being taken as an algorithm parameter.

The path generated by the RRT or the one being optimized by local optimization is a set of points which gives an approximate idea of the trajectory. The trajectory is generated using these points by spline curves [20]. The path points are used as control points to which the output is a smooth and exact path that the vehicle follows. The cost is computed for this smoothed trajectory. The cost of any trajectory is given by equation (4).

$$Cost(\tau) = l(\tau) + c_1 n_1(\tau) + c_2 n_2(\tau) \quad (4)$$

Here $l(\tau)$ is the path length and $n_1(\tau)$ is the number of points in the trajectory which lie at a distance of less than *mindis* from any position of the vehicle. For this factor to be computed the vehicle (extended by a factor of *mindis* from all its sides) is traversed on the trajectory τ and checked for collision. $n_2(\tau)$ is the number of points in the trajectory where the vehicle actually collides with some obstacle on the road, or a point where the kinematic validity constraint as stated in equations (2) and (3) does not hold. This factor is again measured by simulated traversal. c_1 and c_2 are constants. A trajectory with infeasible points cannot be returned, while points with less than the desired clearance distance should be avoided as much as possible. Hence $1 < c_1 << c_2$.

The first term in (4) is introduced to minimize the path length. The second term is to maximize the distance from obstacles (including from other vehicles). The last term is to penalize invalid solutions. The original path formed by RRT is approximate, it is important therefore to use a set of points such that the final trajectory is feasible. As the optimization is local to the range of paths nearby, it may or may not be possible to have a feasible path, and it may or may not be possible to find a path that has sufficient separation from obstacles. The local optimization algorithm cannot and does not search for the path in the global search space as this is computationally costly and as a result extremely time consuming.

Here the problem of global optimization is difficult to solve considering the computational costs of the algorithm. However we also exploit the characteristic nature of the vehicle planning problem which means that that there are limited options due to the narrow width of the road. The length of the un-optimized path returned by the RRT is indicative of the cost for travelling in that area. The RRT algorithm is called a few times and the shortest path length for valid paths is accepted and subsequently further optimized.

The local optimization algorithm is given by Algorithm 3.

Algorithm 3: LocalOptimization(Path)
Best ← Path
BestCost ← Cost(Spline(Path))
repeat for *limit* iterations
    P ← Deviate(Best)
    PC ← Cost(Spline(P))
    if PC<BestCost, BestCost ← PC, Best ← P
end repeat
if BestCost < c_2 return Spline(Best) else return null

3.4 Coordination

Different vehicles need to ensure that they do not come in each other's way and/or collide with each other on their way. This problem is solved by devising a coordination technique between the vehicles. Considering the real time nature of the algorithm, it is not possible to have a centralized or tightly bound technique for coordination. We have therefore chosen a priority based coordination technique [21, 22]. In this, all vehicles are given some priority. The vehicles then plan strictly in the order of their priority. Each vehicle only attempts to avoid collision with vehicles which possess a higher priority and therefore whose plan of action is already precisely known. Lower priority vehicles are not accounted for, they will subsequently need to sort themselves out.

Vehicles are prioritized as per their emergence times into the road segment being planned. A vehicle that emerges first into the planning sequence has a higher priority as compared to a vehicle that arrives later. The vehicles can only see vehicles that lie into the road segment being planned at the time of their emergence and no other vehicle can possibly arrive in that segment in the future – due to the prioritizing. This strategy assures that the plan of vehicles already in the road segment is unaltered as new vehicles arrive later. The arriving vehicles plan so as not to collide with the vehicles already present.

While planning any vehicle the algorithm needs to constantly query the other vehicles possessing higher priority, so as to generate a feasible plan for itself. In this planning the communication is restricted to the vehicle broadcasting a query which is received and replied by all vehicles currently in the road segment being planned. The vehicles return their travel plans to the vehicle being planned. These travel plans are referred to as hashmaps [23] hashed by time which ensures that the vehicle being planned can rapidly know the position of other vehicles at any time.

3.5 Altering Speeds

When computing the path of a vehicle amidst vehicles with higher priorities, there is a choice to either alter the vehicle's path keeping the speed constant or to alter its speed keeping the path constant, or to simultaneously compute both path and speed so as to avoid collision. The RRT search algorithm keeps the vehicle's speed constant and computes the path. If the RRT fails to find a feasible path which is marked by the algorithm failing to reach the goal within the set number of iterations, it may be assumed that no feasible path is possible. Hence the algorithm then needs to alter the vehicle's speed. It is not possible to experiment for a large number of speeds so as to select the highest speed which generates a feasible path due to large computational costs. For similar reasons we cannot reduce the speed by small magnitudes iteratively as was done in [18]. Hence we need to assess the traffic scenario and make a good guess in order to alter vehicle's speed.

In light of the behaviour of an algorithm to a variety of maps as presented in [18], we state that when a higher speed vehicle enters the planning scenario, it sees an array of vehicles in front. These vehicles may be travelling on the same side of the road as the vehicle or on the opposite side. For the array of vehicles travelling on the same side, the vehicle may attempt to go by its preferred speed which, if it is higher than the traversal speeds of vehicles in front, leads to overtaking behaviour. Alternatively it may choose to follow the array of vehicle.

Consider that the vehicle being planned is capable of high speed and that at this speed the vehicle is unable to find a feasible path. In this case overtaking behaviour is not possible. Hence the speed of the vehicle being planned is set to the speed of the higher priority vehicle which is travelling at a lower speed. This causes the vehicle being planned to follow the higher priority vehicle. The alteration goes on iteratively till the speed of the vehicle being planned is reduced to equate to the speed of the slowest vehicle in the array.

However if the vehicle is likely to have a collision with a vehicle travelling in the opposite direction, the task is to alter the speed of the vehicle in such a way that a collision is avoided. For this we have no choice but to iteratively decrease its speed in small steps. In the worst case the speed of the vehicle is reduced to 0 and it stands still. In such a scenario it may prefer to wait for other vehicles leave a clear way. In this way continuous re-planning by small speed changes in pursuit of a feasible path is a valid strategy. If the road is blocked the vehicle may, as a result, wait indefinitely. Hence after some time it may additionally prefer to choose another route, considering the present route to be blocked. We do not deal with this planning in the present version of the algorithm.

Since the higher priority vehicles (if any) travelling in the opposite direction did not themselves account for the presence of a waiting vehicle either at the start or just before the road segment, they may have to validate their paths. It is possible for deadlock to occur in such situations, in which case vehicles must be re-prioritized and re-planned so as to realise a feasible travel plan.

The general planning algorithm is given by Algorithm 4.

Algorithm 4: Plan(road segment, time)

```
while true
    bestPath ← null
    repeat for small number of iterations
        path ← RRT-Connect(current position of v, segment)
        if (bestPath is null ∨ path is of smaller length) ∧ path is feasible,    bestPath ← path
    end repeat
    if bestPlan ≠ null,    trajectory ← LocalOptimize(bestPath)
    if trajectory ≠ null
        represent trajectory as a hash map of time and return trajectory
    else if vmax>0 modify preferred speed vmax
    else stop and wait
    end if
end while
```

4 Results

The algorithm was extensively tested by MATLAB simulation. The algorithm took as input an image file in bmp format which was a black and white pictorial of the road map. The scenario specifying the time of emergence of vehicles, their speeds, positions and orientations was fed separately. Various aspects of the algorithm RRT-Connect, coordination conversions, spline generation, cost computation, etc. were developed as separate modules. The outputs of the algorithm were displayed as an animation which indicated the manner in which the various vehicles entered and moved. To fully test the algorithm a number of experiments were conducted ranging from simple to complex scenarios. Each scenario was selected to display some particular characteristic of the algorithm.

4.1 Single Vehicle Scenarios

We will first discuss here the experiments involving a single vehicle moving on a straight or curved road, with or without obstacles. Various scenarios are shown in Figure 2. Figure 2(a) is the simplest case wherein the vehicle emerges into the centre of the road and sees no obstacles on the road. The only thing it needs to do is to drive along a small curve. The RRT-Connect needed to expand a single node for the scenario and the completion was done by the connect part of the algorithm. The solution was generated in negligible time. The generated solution was feasible and local optimization was in this case unnecessary, though carried out. This shows that for most general driving, algorithmic effort would be minimal, which also displays the present state of planning with speed lanes. Figure 2(b) represents another scenario extensively discussed in the literature. A single obstacle is added which the vehicle needs to avoid optimally. The RRT needed to expand a few nodes until a reasonable node was found after which the connect algorithm could make the vehicle traverse. It may be seen that vehicle keeps to the right side of the road and does not come to the (normal) left side. A couple of iterations of local optimization were needed to rectify the path for feasibility. After a few iterations the exact point of obstacle avoidance could be determined, so that separation from the obstacle was greater than the threshold value.

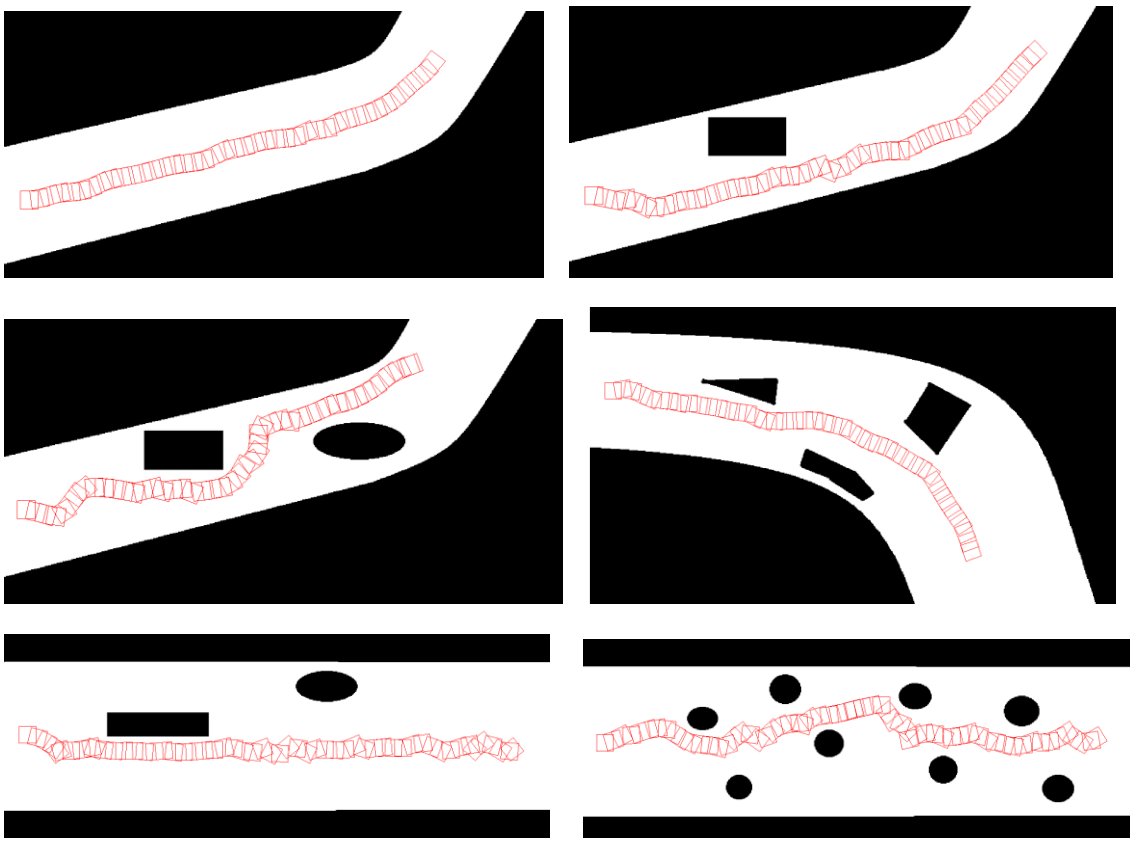

Figure 2: Simple navigation and obstacle avoidance of vehicle.

Figure 2(c) shows a reasonably complex formulation where two obstacles are closely packed and the planner needs to make a fine trajectory to avoid both of them. RRT expansions were large considering the need to try out a large number of possibilities, before one turned out to be feasible. It can be seen that the algorithm could successfully draw out a feasible and optimal trajectory. A similar situation occurred in Figure 2(d). Here obstacles are not tightly packed enabling local optimization to fine tune the path. Figure 2(e) displays a very complex obstacle framework. The algorithm attempts to find a path which is as central as possible, and which also leads to the goal. Considering path smoothness, it is difficult to find points for making turns. Similarly local optimization here has a huge task to do in optimizing the trajectory for minimum separations from the variety of obstacles. Figure 2(f) shows the scenario where two paths were possible. The algorithm chose the simpler and shorter one. It cannot be ascertained that every time the same path is chosen, but the algorithm is certainly more prone to chose the globally optimal path as the path length is approximated by RRT runs and the best are taken.

4.2 Two Vehicle Overtaking and Vehicle Following Scenarios

In the next category of experimentation we used the same maps as before, but employed two vehicles in the scenario rather than a single vehicle. The first vehicle was modelled as a slow moving vehicle which entered the scenario unaware of the second vehicle. It formulated its trajectory and continued its motion. After some time the second vehicle entered the scenario and this was capable of moving at higher speed. The second vehicle hence had a choice to either attempt to overtake the first vehicle or to follow it. It first plans to overtake it, and if a feasible path is not found, it reduces its speed and simply constructs a trajectory by which it follows the first vehicle. Some of the experiments are shown in Figure 3 indicating the scenario a little time after the emergence of the second vehicle and the final trajectories.

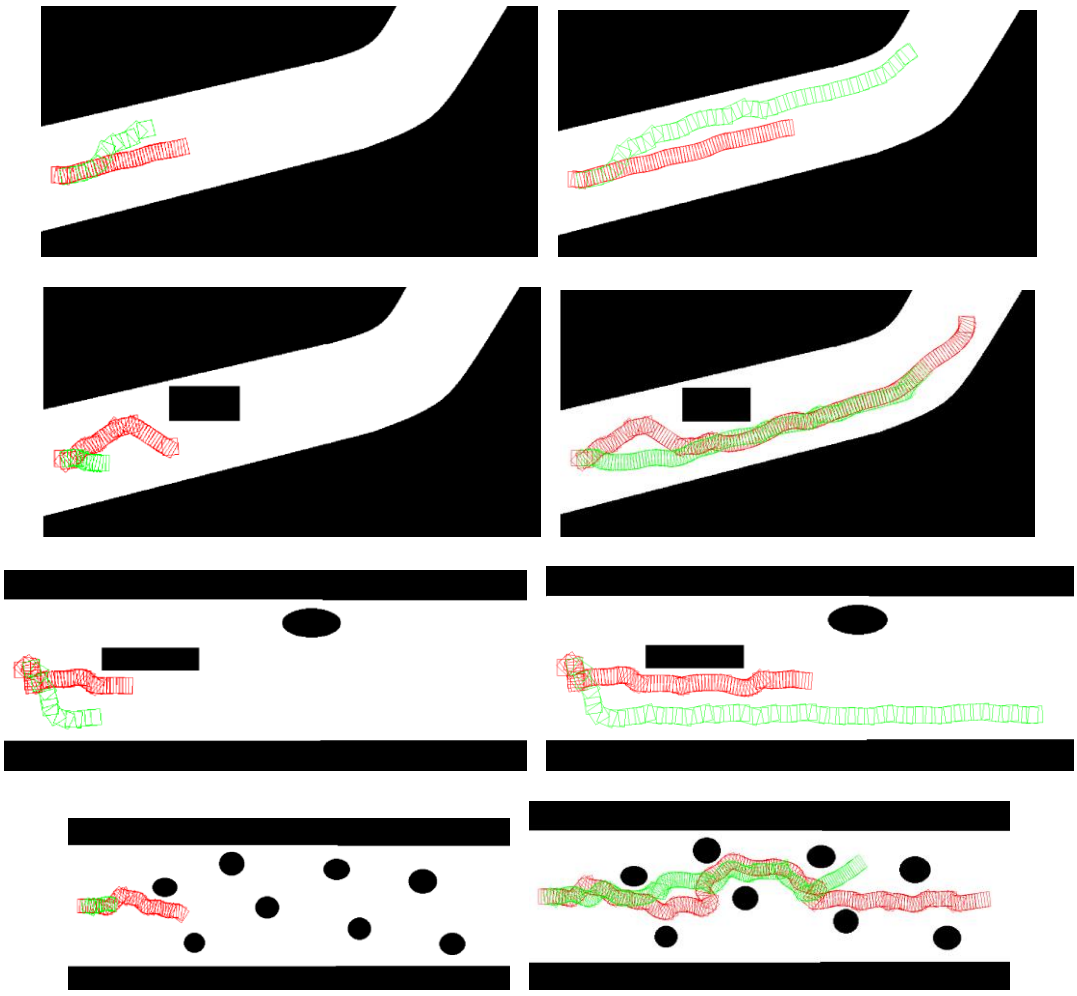

Figure 3. Vehicle following and overtaking behaviour of vehicle

Figure 3(a) shows a road with no obstacle in which the first vehicle was driving in the middle of the road. The second vehicle naturally has a lot of room to simply overtake it which it does as shown in Figure 3(b). The

obstacle landscape for the second vehicle which accounts for the first vehicle is simply a single elongated obstacle, which is easy to overcome. However overtaking in the presence of an obstacle is difficult and can only be done if sufficient space is available. The scenario shown in Figure 3(c) and Figure 3(d) clearly shows that the second vehicle had no space to overtake the first vehicle considering its obstacle landscape is filled up by both a static obstacle and moving vehicle. Hence, as a result, it had to follow the first vehicle.

Figure 3(e) and Figure 3(f) show another scenario in which large spaces were available and hence overtaking was possible. The first vehicle took the straighter route. The second vehicle now had a choice to overtake the first vehicle either on the left or the right. If it does so on the left, it encounters an additional obstacle which it must also avoid. Hence the second vehicle decides to overtake on the right. Both these possible trajectories have roughly the same cost hence a number of experiments resulted in both these paths being selected by the second vehicle on different runs. In the scenarios shown in Figure 3(g) and Figure 3(h), a large amount of space was available, although it was still not large enough for a complete vehicle to fit in and overtake. Both vehicles preferred to remain roughly in the centre of road as per their initial generation. Once the second vehicle decided to follow the first vehicle it reduced its speed and from then on the presence of the first vehicle barely affected the motion of the second.

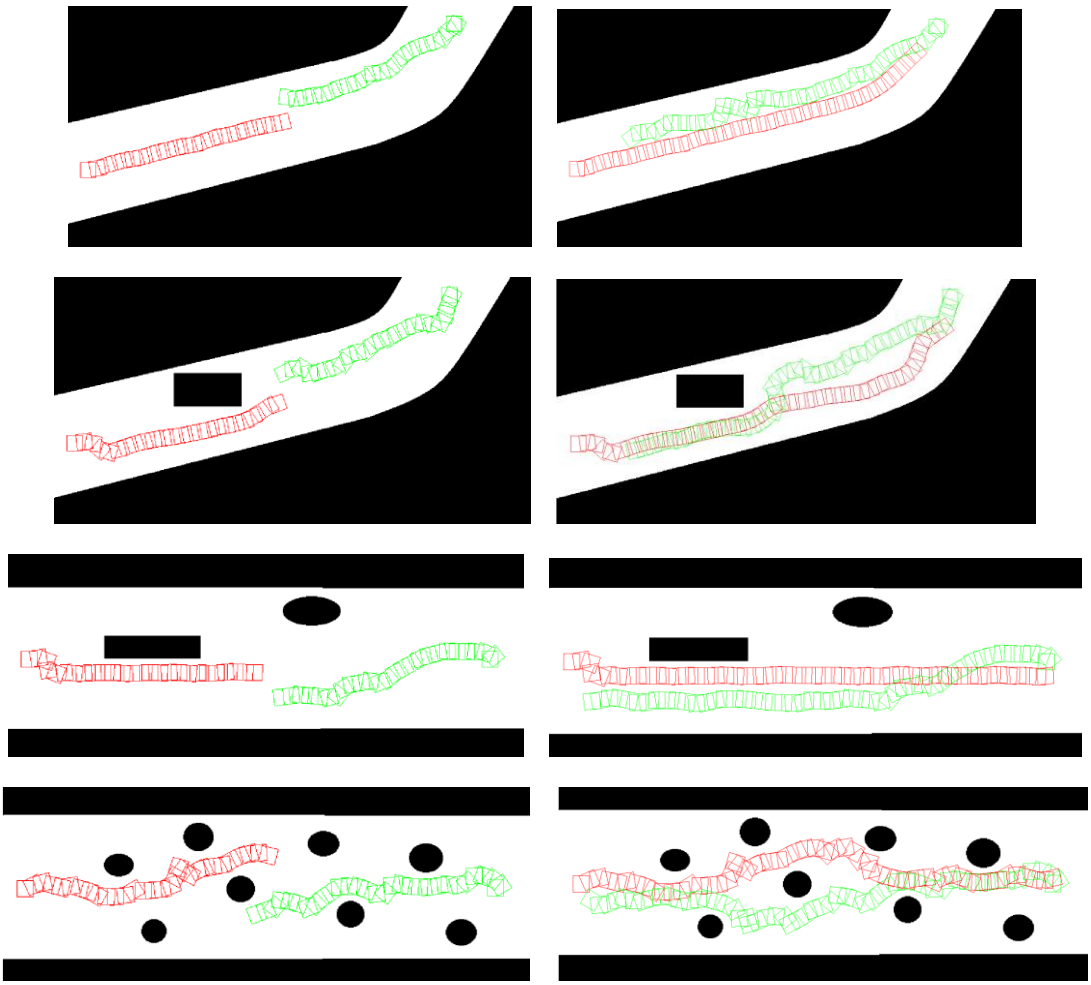

Figure 4: Vehicle avoidance behaviour of vehicle.

4.3 Vehicle Avoidance Scenarios

In all these experiments since the vehicles were assured of no vehicle approaching on the other side of the road, they could make use of the entire road bandwidth. In real situations however the same road would be used for both inbound and outbound traffic which necessitates proper coordination between the two set of vehicles. This ability of the algorithm was also therefore tested via experiments on the same set of maps. In these experimentations two vehicles were generated, one closely after the other, from two opposite ends of the segment and facing each other. The algorithm had to devise trajectories for both these vehicles such that they

avoided each other. This task is twofold. First the algorithm needs to decide what is the highest possible speeds of the vehicles so that any likely point of collision is at a place where sufficient width is available for the vehicles to avoid each other. Then the planning algorithm needs to compute the trajectory. If at some speed the algorithm determines there is likely to be a collision and no feasible path, it reduces the travel speed. This shifts the place where vehicles meet to one where it may be possible to have no collision. If a collision is still planned to occur or no feasible path may be computed, the process repeats. Experiments in a few scenarios are shown in Figure 4. The figures show how the vehicles avoid each other, and henceforth how they complete their journeys.

Figure 4(a) shows the first vehicle travelling straight and hence the second has to make a small turn to avoid a collision. The planning of the second vehicle is also simple since once the collision avoidance point is found, the RRT-Connect algorithm proceeds to complete the trajectory. The vehicle which enters first gets the straighter path (in the road coordinate axis system). By the choice of speeds which were kept equal, the scenario shown in Figure 4(c) and 4(d) was simple. The vehicles meet at a point where the second vehicle has a lot of space to modify its trajectory so as to avoid collision. Planning for the second vehicle is however slightly difficult as it first has to avoid the vehicle and then the static obstacle. Similar comments hold for the scenario shown in Figure 4(e) and 4(f). The scenario shown in Figure 4(g) and 4(h) is more complicated however as the second vehicle has to not only escape from the static obstacle framework which is itself complicated, but also avoid the first vehicle. This makes the obstacle landscape very complex. It may be seen how the vehicles can coordinate to travel on opposite sides of the obstacles so as to avoid collision.

4.4 RRT Analysis

It is important for RRT-Connect to generate a trajectory within a small execution time. Local optimizations are an optional feature which may be timed to assess their performance. This means that it must be possible to reach the goal by means of minimal expansions of the tree even in complex environments. The RRT generated for a static scenario is shown in Figure 5(a). It can be seen that effectively the algorithm only generates RRT until a point from which a straight path (in the road coordinate axis system) will reach the goal. Hence complexity is proportional to the number of obstacles. Since, in usual driving, the number of obstacles is low, the algorithm is likely to generate paths with small overall execution times. Further even with a high number of obstacles, the algorithm tries to restrict itself to good expansions, rather than expanding the entire road available. The path found for the map is shown in Figure 5(b), which after local optimizations results in the trajectory as shown in Figure 5(c).

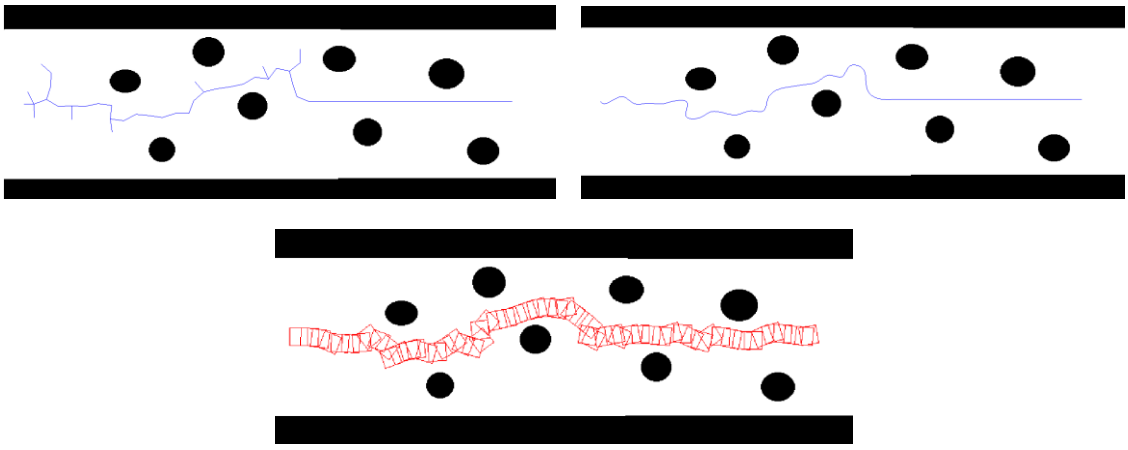

Figure 5: Generation of RRT and trajectory for single vehicle.

The scenario is more interesting in the presence of other vehicles, whose planned trajectory is known and which the algorithm attempts to cause the vehicle to avoid. The same set of figures with a single vehicle is shown in Figure 6. The figure shows a case in which space was available and the vehicle could overtake. Figure 6(a) shows a region in between the red trajectory of the slow vehicle already planned and the green RRT generated which appears as an obstacle area. It is the region where the first vehicle's motion as per its space time results in a collision with the second vehicle, which hence had to take a longer route to avoid a collision. It is worth noting that even in the presence of other vehicles the number of nodes expanded is fairly low which means that computation time is minimal. The curve generated for two vehicles is shown in Figure 6(b).The final trajectory after local optimization is shown in Figure 6(c).

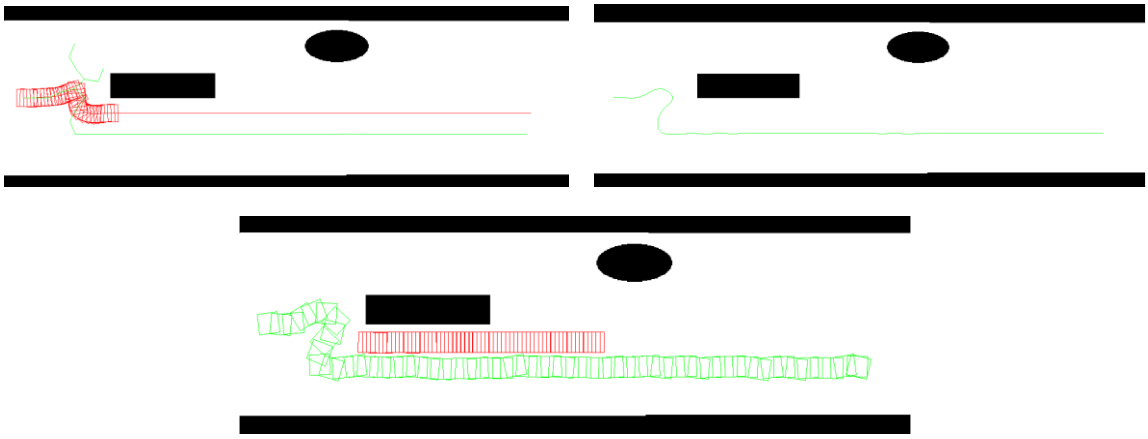

Figure 6: Generation of RRT and trajectory in case of multiple vehicles.

4.5 Local Optimization Analysis

The other important aspect is to analyze the local optimization performed by the algorithm. Initially the RRT generates a plan which ensures that the vehicle would not collide with any obstacle when placed at the nodal points. However in a continuous domain where the path is smoothed by splines, there may be collisions. This is therefore the first task performed by the local optimization algorithm. Then the local optimization tries to increase separation distances so that the vehicle's distance from all obstacles is greater than the specified threshold. Following this the algorithm attempts to minimize the path length as the last criterion, which is considered to be of least importance. This is shown in Figure 7(a) and 7(b) for optimization on a random scenario with only static obstacles, and Figure 8(a) and 8(b) for optimization with an additional vehicle. In both these cases it may be seen that the path returned by the RRT-Connect algorithm was actually infeasible, but could be rectified in a single iteration. The distance was slowly increased and later the path was optimized in both these approaches. The deviation factor accounting for the magnitude of deviation was kept a little high as compared to other approaches. This was done to ensure that the algorithm quickly made a feasible path, with all other objectives being secondary. This ensured that relatively few iterations of the local optimization routine were sufficient for trajectory generation.

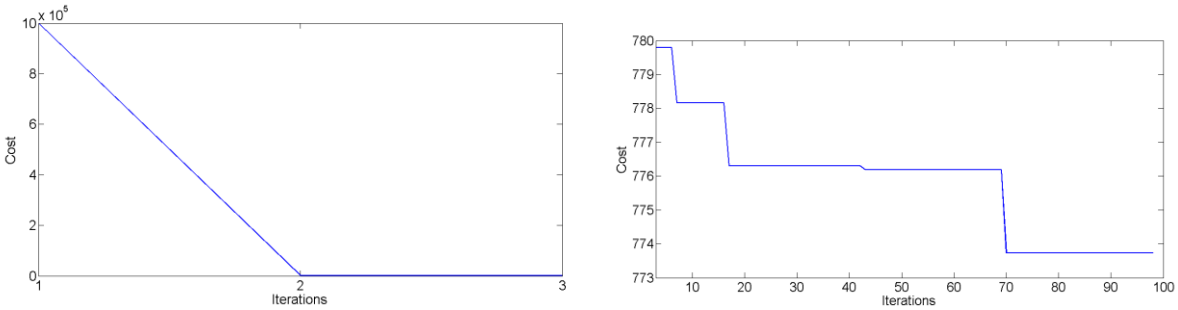

Figure 7: Performance of Local Optimization Algorithm for Single Vehicle

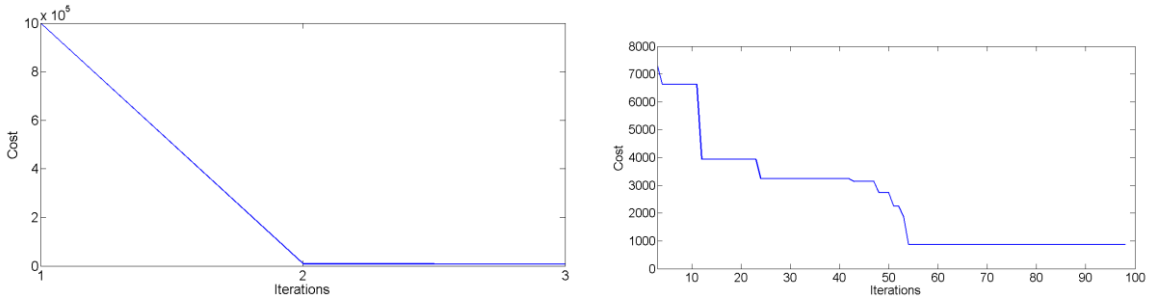

Figure 8: Performance of local optimization algorithm in presence of multiple vehicles.

5 Related Works

The non-assumption of speed lanes makes the problem of planning multiple autonomous vehicles equivalent to mobile robot motion planning for which RRT is extensively used. In a recent work Raveh et al. [24] state how multiple independent instances of a sampling based algorithm may be combined using a dynamic programming approach. The resultant algorithm combines best path segments from different instances of sampling the algorithm to make the best overall path for the robot. The merging algorithm, being dynamic programming, is fast. The technique is however suited to only single robot applications. The harder part of the problem targeted by the authors was to make a robot select the strategy to avoid an obstacle. However for road scenarios consisting of few obstacles and more open spaces, local optimization was better suited.

For planning with multiple robots Kala [25] used co-evolutionary genetic programming in which each robot's path was a genetic programming instance. Multiple such instances (corresponding to the various robots) were coordinated for the generation of the optimal travel plan. The master genetic algorithm had the task of selecting the best genetic programming instances of each robot such that the total path was optimal. The author modelled a robot to wait for another robot and used a path segment from memory which was common between the robots. However, being an evolutionary technique, the algorithm was too slow to work within small execution times. Further the solution developed was to navigate through known pathways for multiple robots, rather than avoiding complex obstacles. In other words obstacles were not considered.

An application of real time robot motion planning can be found in the work of Bruce and Veloco [26]. In this approach the authors presented both a planning mechanism and robotic control. While planning carried the task of building a general layout of the motion of robot, the control mechanism assured motion under the constraints of system dynamics. The system also used a random search mechanism for assuring that the robot was always within the permissible limits and any potential possibility of velocity overshoot or undershoot was controlled in a timely fashion. The planning algorithm implemented Execution Extended Randomly-exploring Random Trees (E-RRT) for planning. In this few of the previous states of the selected paths were stored in cache, which affected all future exploration of the system. In this manner a generation was able to pass winning characteristics to the next generation. In our approach we prefer to keep safety separations from all obstacles and vehicles so that the vehicle may be easily controlled without requiring re-planning. This is better for our scenario as the environment is known whilst environment dynamics are not too uncertain.

A problem associated with RRTs is their performance in conditions like thin corridors. This is due to the fact that it is reasonably hard to get points inside the narrow corridor which ultimately lead the robot to take an alternative route. Zhang and Manocha [27] used retraction based RRT planner where the samples within the obstacles would be promoted to the nearest possible point on the surface. This resulted in the generation of a good number of samples that could ensure effective planning. The problem of study however was the narrow corridor problem which is absent in road scenarios. This work may however be extended using the authors' sampling technique for better performance in such cases. In all other respects the solution is equivalent to RRT.

Planning in the presence of speed lanes is simpler and mainly deals with the choice of speed lane of travel at any time. Sewall et al. [28] studied the problem of traffic reconstruction and used the A* algorithm for the planning of individual vehicles in the presence of speed lanes. Vehicles were prioritized as per their position in a road segment and planned strictly by their priorities. Planning involved deciding the optimal change of speed lane. The authors attempted to plan so as to reduce the total number of lane changes, reduce the total acceleration, and attempted to maximize the separation between vehicles. Schubert et al. [29] presented a framework wherein the vehicle could detect the speed lanes on roads. Decisions were made on the basis of separation with vehicles directly ahead of the vehicle and directly to the rear of the vehicle. This was converted into probabilities for driving in various lanes. Bayesian networks were then used to make the final decision.

Similarly Furda and Vlacic [30] broke down the planning decision into higher and lower level decisions. Higher order decisions related to the choice of speed lanes which were made as per knowledge of the speeds and distances of other vehicles by a multi-criterion decision making technique. Lower order decisions dealt with trajectory generation for changing speed lanes or sticking to the same speed lane. These approaches modelled similar sized vehicles with small or no differences in preferable driving speeds. No cooperation mechanism between the vehicles was modelled however. Each vehicle only attempted to maximize its own goal. Decision making was restricted to discrete decisions regarding choosing the speed lane of travel.

Overtaking with the assumption of speed lanes is a well studied problem and usually involves a set of speed lane changes, making an overtaking manoeuvre. Naranjo et al. [31] used a fuzzy rule based system for constructing

and controlling the vehicle for all speed lane changes. Here each lane change was guided by a different set of rules. Lateral distances of the vehicle were used as fuzzy inputs. Hegeman et al. [32] meanwhile developed an assistance system for drivers to help them in decision making regarding overtake. The assistance system used distances and speeds of all vehicles to decide the feasibility of overtaking. The driver could be allowed to initiate overtake if the probability of collision was determined to be below a threshold. Overtaking in the presence of speed lanes can be simply modelled as changing speed lanes. Since traffic is only going from one side to the other, vehicle motion is feasible if lane changes are feasible. Modelling in either [31] or [32] did not include the possibility of any additional vehicle interfering with the overtaking. The decision is much harder to predict in a non-speed lane scenario where the same road is used for both inbound and outbound traffic.

6 Discussion

Speed lanes are a defining restriction that is useful for vehicles driven by humans. They are not necessary for robotic vehicles other than to simplify the problem. However in doing so any solution naturally becomes non-optimal.

It is essential to model the absence of speed lanes in a number of scenarios especially where traffic is prone to be diverse. Planning without speed lanes enables every vehicle to go by its own way, which increases the problem complexity from a planning perspective. It is also important to have the planning algorithm terminate with a feasible travel plan within a small threshold of time. This ensures that any changes to the environment from the moment of triggering the planning algorithm to when a solution is returned and acted upon is minimal and can be neglected. Further it ensures that the vehicles themselves don't move appreciably while the plan is being made, so as to affect the feasibility of generated plan. The trade off between optimality and completeness of the algorithm to the execution time is an important consideration when choosing a planning algorithm. The choice largely depends upon the scenario or the application domain.

In the specific domain of planning autonomous vehicles on a road we presented here a variety of practical scenarios. Considering the increasing autonomy in vehicles, it is likely that we will see vehicles talking to each other to decide how to cooperatively navigate in the presence of obstacles. Diversity amongst vehicles is then a potential source of large dynamics in vehicle behaviour planning and operation.

In the present approach we have used the RRT-Connect algorithm to plan an individual vehicle with priority based coordination amongst vehicles. This scheme is able to generate feasible plans in small execution times for a diverse set of scenarios. Vehicles can plan their trajectories as well their speed of motion, considering the presence of other vehicles. The algorithm addresses a large number of questions ensuring the generation of a good, safe and timely plan for multiple vehicles.

In the present version however cooperation is weakly implemented. A slower vehicle is unable to offer extra space for a faster vehicle to overtake. Further higher priority vehicles always take better paths, rather than cooperating to enable better trajectories for slower vehicles. The current approach also operates a failsafe speed in which a faster vehicle decides to follow a slower vehicle rather than attempting to overtake at a different speed, or trying to bargain about the highest speed to follow. The local optimization algorithm used did a good job. However the optimization landscape is neither uni-modal, nor low in dimensionality. Making a path deviation adaptive may result in better optimization or alternatively an effective but fast optimization scheme may be defined. Naturally in time the algorithm needs to be tested on physical vehicles on physical roads.

7. Conclusions

In this paper we presented a method to plan multiple autonomous vehicles in the absence of speed lanes. The intent is to generate a collaborative collision-free travel plan involving all vehicles on the road. The developed solution makes use of priority based coordination between vehicles. Each vehicle plans using RRT-Connect and local optimization. In case a vehicle is unable to generate a feasible trajectory considering the motions of higher priority vehicles, it may need to reduce its speed and re-plan. In order to generate a feasible plan within a small execution time, the algorithm attempts to make the vehicle follow the vehicle in front if one exists, failing which its speed is reduced by a fixed magnitude. The RRT-Connect algorithm ensures that feasible trajectories are generated by minimal expansions, the result of this being that algorithm execution times are low.

The algorithm was experimented on with a variety of scenarios ranging from simple to difficult. The purpose was to test whether the algorithm is able to generate a feasible trajectory even for scenarios where it may be

difficult to manoeuvre the vehicle. This ensures that the algorithm is scalable to complex situations whilst at the same time returning solutions within a small execution time threshold. The results showed that a vehicle could be easily navigated with plans returned reasonably early. The plans could be locally optimized using few iterations of a local planner for shorter and safer paths. A slower vehicle could always overtake a faster vehicle, if space was available. Further vehicles could mutually avoid each other.

Acknowledgement

The authors wish to thank the Commonwealth Scholarship Commission in the United Kingdom and the British Council for their support of the first named author through the Commonwealth Scholarship and Fellowship Program - 2010 - UK award number INCS-2010-161.

References

[1] M. Buehler, K. Iagnemma, S. Singh, The 2005 DARPA grand challenge: the great robot race, Springer, Berlin, Heidelberg, 2007.

[2] G. Seetharaman, A. Lakhotia, E. P. Blasch, Unmanned Vehicles Come of Age: The DARPA Grand Challenge, Computer, 39(12) (2006), 26-29.

[3] S. Tsugawa, Inter-vehicle communications and their applications to intelligent vehicles: an overview, Proceedings of the IEEE Intelligent Vehicle Symposium Vol. 2, 2002, 564- 569.

[4] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt, S. Thrun, Junior: The Stanford entry in the Urban Challenge, Journal of Field Robotics, 25(9)(2008), pp 569–597, 2008.

[5] S. A. Nobe, F. Y. Wang, An overview of recent developments in automated lateral and longitudinal vehicle controls, Proceedings of the 2001 IEEE International Conference on Systems, Man, and Cybernetics vol.5, 2001, 3447-3452.

[6] L. Vanajakshi, S. C. Subramanian, R. Sivanandan, Travel time prediction under heterogeneous traffic conditions using global positioning system data from buses, IET Intelligent Transport Systems, 3(1) (2009), 1-9.

[7] J. R. Alvarez-Sanchez, F. de la Paz Lopez, J. M. C. Troncoso, D. de Santos Sierra, Reactive navigation in real environments using partial center of area method, Robotics and Autonomous Systems, 58(12) (2010), 1231-1237.

[8] R. Kala, A. Shukla, R. Tiwari Fusion of probabilistic A* algorithm and fuzzy inference system for robotic path planning, Artificial Intelligence Review, 33(4) (2010), 275-306.

[9] O. Khatib, Real-Time Obstacle Avoidance for Manipulators and Mobile Robots, The International Journal of Robotics Research, 5(1986), 90-98.

[10] R. Kala, A. Shukla, R. Tiwari Robotic Path Planning using Evolutionary Momentum based Exploration, Journal of Experimental and Theoretical Artificial Intelligence, 23(4) (2011a), 469-495.

[11] R. Kala, A. Shukla, R. Tiwari, Dynamic Environment Robot Path Planning using Hierarchical Evolutionary Algorithms, Cybernetics and Systems, 41(6) (2010), 435-454.

[12] R. Kala, A. Shukla, R. Tiwari, Robotic path planning in static environment using hierarchical multi-neuron heuristic search and probability based fitness, Neurocomputing 74(14-15) (2011b), 2314-2335.

[13] J. J. Kuffner, S. M. LaValle, RRT-connect: An efficient approach to single-query path planning, Proceedings of the IEEE International Conference on Robotics and Automation vol. 2, 2000, 995-1001.

[14] S. M. LaValle, J. J. Kuffner, Randomized kinodynamic planning, Proceedings of the IEEE International Conference on Robotics and Automation, 1999, pp. 473–479.

[15] Y. Kuwata, S. Karaman, J. Teo, E. Frazzoli, J. P. How, G. Fiore, Real-Time Motion Planning With Applications to Autonomous Urban Driving, IEEE Transactions on Control Systems Technology, 17(5) (2009), 1105-1118.

[16] J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman, O. Koch, Y. Kuwata, D. Moore, E. Olson, S. Peters, J. Teo, R. Truax, M. Walter, D. Barrett, A. Epstein, K. Maheloni, K. Moyer, T. Jones, R. Buckley, M. Antone, R. Galejs, S. Krishnamurthy, J. Williams, A perception driven autonomous urban vehicle, Journal of Field Robotics, 25(10) (2008), 727–774.

[17] S. Chakravorty, S. Kumar, Generalized sampling based motion planners with application to nonholonomic systems, Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, 2009, 4077-4082.

[18] R. Kala, K. Warwick, Planning of Multiple Autonomous Vehicles using RRT, Proceedings of the 10th IEEE International Conference on Cybernetic Intelligent Systems, Docklands, London, 2011.

[19] E. K. Xidias, P. N. Azariadis, Mission design for a group of autonomous guided vehicles, Robotics and Autonomous Systems, 59(1) (2011), 34-43.

[20] C. de Boor, A Practical Guide to Splines, Springer, Berlin-Heidelberg, 1978.

[21] M. Bennewitz, W. Burgard, S. Thrun, Optimizing schedules for prioritized path planning of multi-robot systems, Proceedings of the 2001 IEEE International Conference on Robotics and Automation, Seoul, Korea, 2001, 271 – 276.

[22] M. Bennewitz, W. Burgard, S. Thrun, Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots, Robotics and Autonomous Sysems, 41(2-3) (2002), 89–99.

[23] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Chapter 11: Hash Tables, Introduction to Algorithms, Second Edition, MIT Press, Massacusettes, 2001, 221-245.

[24] B. Raveh, A. Enosh, D. Halperin, A Little More, a Lot Better: Improving Path Quality by a Path-Merging Algorithm, IEEE Transactions on Robotics, 27(2) (2011), 365-371.

[25] R. Kala, Multi-Robot Path Planning using Co-Evolutionary Genetic Programming, Expert Systems With Applications, 39(3) (2012), 3817-3831.

[26] J. Bruce, M. Veloso, Real-Time Multi-Robot Motion Planning with Safe Dynamics, In: Multi-Robot Systems: From Swarms to Intelligent Automata Vol. 3, Springer, Heidelberg, 2005, 159-170.

[27] L. Zhang, D. Manocha, An efficient retraction-based RRT planner, IEEE International Conference on Robotics and Automation, 2008 2008, 3743-3750.

[28] J. Sewall, J. van den Berg, M. C. Lin, D. Manocha, Virtualized Traffic: Reconstructing Traffic Flows from Discrete Spatio-temporal Data, IEEE Transactions on Visualization and Computer Graphics, 17(1) (2011), 26-37.

[29] R. Schubert, K. Schulze, G. Wanielik, Situation Assessment for Automatic Lane-Change Maneuvers, IEEE Transactions on Intelligent Trasportation Systems, 11(3) (2010), 607-616.

[30] A. Furda, L. Vlacic, Enabling Safe Autonomous Driving in Real-World City Traffic Using Multiple Criteria Decision Making, IEEE Intelligent Trasportation Systems Magazine, 3(1) (2011), 4-17.

[31] J. E. Naranjo, C. González, R. García, T. de Pedro, Lane-Change Fuzzy Control in Autonomous Vehicles for the Overtaking Maneuver, IEEE Transactions on Intelligent Trasportation Systems, 9(3) (2008), 438-450.

[32] G. Hegeman, A. Tapani, S. Hoogendoorn, Overtaking assistant assessment using traffic simulation, Transportation Research Part C, 17(6) (2009), 617–630.