Heuristic based Evolution for the Coordination of Autonomous Vehicles in the Absence of Speed Lanes

Rahul Kala and Kevin Warwick
School of Systems Engineering, University of Reading,
Whiteknights, Reading, RG6 6AY, UK
rkala001@gmail.com, k.warwick@reading.ac.uk
Ph.: +44 - 7424752843

Citation: R. Kala, K. Warwick (2014) Heuristic based evolution for the coordination of autonomous vehicles in the absence of speed lanes. Applied Soft Computing, 19: 387–402

Final Version Available At: http://www.sciencedirect.com/science/article/pii/S1568494613003608

Abstract
The current state of the art in the planning and coordination of autonomous vehicles is based upon the presence of speed lanes. In a traffic scenario where there is a large diversity between vehicles the removal of speed lanes can generate a significantly higher traffic bandwidth. Vehicle navigation in such unorganized traffic is considered. An evolutionary based trajectory planning technique has the advantages of making driving efficient and safe, however it also has to surpass the hurdle of computational cost. In this paper we propose a real time genetic algorithm with Bezier curves for trajectory planning. The main contribution is the integration of vehicle following and overtaking behaviour for general traffic as heuristics for the coordination between vehicles. The resultant coordination strategy is fast and near-optimal. As the vehicles move, uncertainties may arise which are constantly adapted to, and may even lead to either the cancellation of an overtaking procedure or the initiation of one. Higher level planning is performed by Dijkstra's algorithm which indicates the route to be followed by the vehicle in a road network. Re-planning is carried out when a road blockage or obstacle is detected. Experimental results confirm the success of the algorithm subject to optimal high and low-level planning, re-planning and overtaking.

Keywords: Unmanned Ground Vehicles, Autonomous Vehicles, Traffic Simulation, Multi-Robot Systems, Multi-Robot Coordination, Motion Planning, Robotics

1. Introduction

Since the initiatives of the DARPA Urban Challenge [1] there has been a considerable increase in the interest in the control of autonomous vehicles in traffic scenarios. Engineering the complete vehicle requires decisions to be made over the types of sensors to be employed, their location, sensor data pre-processing and processing, sensor fusion, map building, motion planning, motion control, etc [2-3]. While these vehicles are able to steer themselves efficiently in the correct manner, less attention is however paid to devising mechanisms for their coordination. That said, advances in Multi-Robot Systems [4-5] present possibilities for multiple robot solutions to such a problem by means of task division and mutual coordination. With the basic assumption that the various robots involved have an open means of communication with each other – either directly or through a common server - algorithms may be devised to ensure the development of an overall optimal strategy.

The basic problem to be dealt with is to move a number of vehicles from their start point to their destination, along the roads which are known apriori. The problem can be seen as a multi-robot path planning problem [6-7]. Approaches taken to solve these problems can be centralized or decentralized [8], however it is clear that decentralized approaches are preferable for most real time and uncertain environments.

The presence of speed lanes is a well-accepted notion in the domain of intelligent vehicles and intelligent transportation systems. Associated planning algorithms generally involve a decision as to the optimal lane of travel and manoeuvring vehicles to their desired speed lane. To the best of our knowledge, there is no significant work on the planning of vehicles in the absence of lanes. This work carries out the task of planning in such a scenario. Further, the presence of different lanes for incoming and outbound traffic is not assumed. The resultant traffic system is hence unorganized where vehicles may have any desired lateral position on the road.

Unorganized traffic can lead to a higher traffic bandwidth in traffic scenarios where vehicles differ largely in their widths and can enable close overtaking. Lane width is computed based on the largest width vehicle which

may occupy the road. In a scenario where the traffic consists of a large number of vehicles with smaller widths, the lanes they drive on therefore have a part of the lane underutilized. The underutilized parts of lanes across the width of the road can be used to accommodate extra vehicles, which is only possible if lanes are not followed or the traffic is unorganized. Further, not adhering to lanes means that overtaking can be completed by pushing the other vehicles to the side to get some additional overtaking space. This may result in infeasible overtaking (as per lane based travel) being made feasible and overtaking being performed as early as possible. The advantages of close overtaking are realizable in traffic scenarios where vehicles differ largely in their travel speeds.

The advantages of unorganized traffic are demonstrated by means of two examples. Lane splitting (where the motorists drive in-between lanes) is allowed at a number of places where the motorists defy lanes resulting in a better travel plan for themselves, which does not significantly affect other vehicles. Sewall et al. [9] specifically state incorporating lane splitting as one of the future directions of their work. In this paper however we have generalized the notion further to incorporate partial and overlapping lane occupancy by any vehicle or even the removal of lanes all together. The basis for this is that many countries follow unorganized traffic where lanes are not utilized and traffic generally shows a higher bandwidth with diverse vehicles. Take the Indian traffic as an example. The size and speed diversity includes cycles, motor bicycles, 3-wheeled auto rickshaws, cars, buses and trucks. In such cases, smaller vehicles constitute a larger proportion of the traffic. It is common to see vehicles nicely fitting in-between others so as to occupy the entire road resulting in a higher bandwidth. Constant overtaking is common.

Organized traffic, where lanes are followed, has the advantages of a higher degree of safety, clearer intentions of other vehicles and fewer lane changes or lateral movements which result in a more comfortable driving experience and possibly shorter travel distances. Unorganized traffic is more risky and continuous care must be taken as the adjacent vehicles may manoeuvre in a variety of ways which can cause uncertainty [10]. Hence when most of the vehicles are roughly wide enough to occupy the entire lane and the vehicles have similar preferred speeds, organized traffic is a better choice. However as speeds and sizes of vehicles become more diverse so the advantages of unorganised traffic become more important.

Not considering lanes necessarily increases the problem of planning and coordination to a large level but provides for a more general solution. The most important problem faced here is in overtaking, due to the requirement for safety under optimal steering conditions, resulting in a small margin for error in efficiency of driving. However, the vehicle may ask other vehicles for support in order to make a close overtake possible. Such a solution is inspired from observed driving in low width high density roads, especially where vehicles vary in speeds to a large degree – English country lanes are a good example!

Whilst driving along straight roads may be a relatively straightforward affair, making efficient turns for most crossings, or other natural turnings of the road, requires expertise. Turning too close to the inner boundary may require reducing the speed, at the same time turning along the outer boundary may make the path too long and sub-optimal. In this paper we try to do the same by using a Bezier based motion planning linked with a genetic algorithm for the optimization procedure. The genetic algorithm is adapted to work on real time scenarios by using a space-time approach modelling of different vehicles and a global referenced individual representation. A genetic algorithm is probabilistically optimal and probabilistically complete and can work in continuous spaces. The objectives of efficiency and safety can be easily knit to a single objective function for evolutionary optimization. Road scenarios have limited possible homotopies for a vehicle, which translates to a limited modality of the fitness landscape (given the fact that a repair operator is designed which maps every possible trajectory to a reasonable looking, feasible and short trajectory). For a limited modality fitness landscape, the optimal solution can be found reasonably early or, to put it another way, the probability of finding the optimal solution is high. Here assumptions are made on a limited number of obstacles and other vehicles, and with a limited resolution map.

The algorithm is a real life application of dynamic evolutionary algorithms, which optimize an objective under a changing fitness landscape. The operational scenario continuously changes as the vehicles move. At every instant the latest instance of scenario is given to the genetic algorithm for optimization. A single population pool of the genetic algorithm is maintained for a vehicle, which adapts to the changing scenario. This is an example of incremental learning where the learnt model (genetic algorithm population) has to be incrementally updated against the changing data (scenario). The changes may be gradual requiring the genetic algorithm to incrementally learn by small changes to the model (genetic algorithm population), or the changes may be large effectively requiring re-working of the complete model. Enabling optimization to be carried out in real time involves a variety of challenges, which are handled by various methods in this manuscript. As the solution is

designed for a specific application, deterministically adaptive strategies can be framed for each of the challenges.

Traffic rules of everyday driving can in fact play a major role in coordinating the motion of multiple vehicles in general scenarios. Rules such as driving on the left (or right) hand side of the road, and overtaking on the right (or left) play a major role in enabling drivers to plan their motion amidst multiple vehicles. While tackling the complete problem of vehicle coordination would be extremely large, we observe that embedding these rules as heuristics can play a major role in realising an overall efficient strategy. The use of other means, such as priority based planning [11-12], co-evolutionary genetic algorithms [13-14], or any other related technique (see e.g. [15-16]) would either lead to a sub-optimal solution or would require highly computational procedures which are presently not possible in a distributed environment.

In the domain of intelligent vehicles, while most works consider the notion of speed lanes, a few algorithms are capable of working in the absence of speed lanes. Kuwata et al. [17] used Rapidly-exploring Random Trees (RRT) for the planning of a single autonomous vehicle. The work considered other vehicles as obstacles and coordination was non-cooperative, which is a major limitation. Kala and Warwick [18] however used the RRT-Connect algorithm to solve the problem for planning multiple autonomous vehicles with communication. Priority based coordination was used which is non-cooperative. The algorithm can operate using low computational time, however due to the highly sub-optimal nature of the RRT algorithm it can be possible to miss out on a possibility to overtake a vehicle, and instead to follow it. The algorithm does not adapt the plans on-line which means that even if the overtake later becomes possible, it would not be performed. On the contrary, a poor control mechanism for an overtaking trajectory can lead to collisions. Not only is the proposed algorithm better in terms of optimality, online adaptations enable judicious decision making when situations change.

The algorithms used in robot motion planning may however be of use for planning vehicles in the absence of speed lanes, and hence a few related approaches are presented. Multi-layered planning is fundamentally used in the domain of robotics [19-21]. Kala et al. [22] fused probabilistic A* with the Fuzzy Inference System, realising the mixed advantages of the constituent algorithms. Similarly Xiao et al. [23] used an offline planner for static obstacles and an online planner for dynamic obstacles, where preliminary offline planning can be a disadvantage. Due to their real time nature of operation, potential based approaches are extensively used for motion planning, and especially obstacle avoidance [24]. Fahimi et al. [25] used a harmonic potential field with concepts of fluid dynamics for the motion planning of multiple mobile robots. All these approaches cannot generate vehicle following and cooperative overtaking behaviours. They are either sub-optimal or computationally expensive. Unlike these approaches, segments of maps are taken in place of entire maps in our approach.

A large amount of work exists for robot motion planning using Genetic Algorithm (GA). Xidias and Azariadis [26] used GA to solve the vehicle routing and trajectory planning problem. The authors embedded all travel information of all vehicles into one genetic chromosome. Minimizing their fitness function represented an attempt to minimize the maximum travel time of any vehicle. Kala et al. [27] used a two level planning technique at finer and coarser levels, using GA at both levels. The authors enabled the algorithm to work with high resolution graphs in dynamic environments as the breakup of resolution at both levels ensured only a small computational load. The greatest limitation of the authors approach was un-cooperative behaviour in the presence of other robots. Further, in both approaches, the entire planning was offline, plans were non-adaptive, and the use of a conventional system for the representation of points made the algorithm computationally intensive.

Another relevant use of evolutionary computing can be found in the work of Garcia et al. [28] who modify Ant Colony Optimization by using a distance to goal measure in computing the node's cost. This accelerates the algorithm convergence. A memory unit is proposed to track the regions which have been explored and further exploration is resisted. The algorithm however works for discrete spaces. Lepetic et al. [29] solved the problem of computing the trajectory for a mobile robot in robot soccer which requires real time planning. The authors optimized spline curves. A number of scenarios were pre-computed and stored relative to the robot, while the intermediate scenarios could be interpolated. Pre-computation is not possible in traffic related systems. A slight change in orientation which changes the possibility of an overtake can mean an entirely different plan, or the impossibility of a vehicle to go further due to limited road width.

Schubert et al. [30] proposed a method for vehicle decision making in a traffic scenario. The authors computed the probabilities to make a lane change left, lane change right, or to stick to one's own lane. Bayesian networks

were used to make the final decision. Similarly Furda and Vlacic [31] dealt with both higher and lower order decision making regarding the motion of the vehicle. Higher order rules dealt with decisions regarding lane changes. Lower order decisions enabled a vehicle to remain in a lane and maneuver as per requirements. In both these approaches, the assumption of speed lanes and the assumption of having inbound and outbound traffic are major limitations of the work.

A lot of work has been done to use fuzzy controllers for overtaking. Naranjo et al. [32] described the manoeuvre to be composed of three stages - first lane change to the left lane, circulation in the left lane, and second lane change to the right lane, each with its own rule set. Jin-ying et al. [33] divided the entire behaviour into different segments which saw the vehicle making lane change, overtaking, and returning to the original lane. All these became membership functions to a rule based fuzzy controller. Onieva et al. [34] optimize fuzzy controller using iterative GA. All of these fuzzy approaches only dealt with motion of the vehicle and not the decision making regarding whether or not to overtake and if so when and how, which is a prime goal in this paper.

Hegeman et al. [35] developed an assistance system for drivers which displayed how safe it is to overtake. On exceeding the safety threshold, a vehicle was allowed to overtake by making any overtaking manoeuvre. Further Wang et al. [36] used an uncertainty model of a vehicle to compute the probability of a collision between vehicles, if and when an overtaking procedure was to take place. In both approaches however there are considerable limitations in that the decisions are binary, and once a wrong decision has been made, vehicles cannot adapt. Further, the presence of oncoming vehicles, which was not accounted for in these works, puts a strict threshold on the maximum duration within which an overtake must happen, otherwise the oncoming vehicle and the vehicle attempting to overtake would collide.

The approach developed here for solving the multi-vehicle coordination problem (see Figure 1) consists of Dijkstra's algorithm for path selection (section 2.1), this enables the vehicle to reach its goal from the source. The algorithm works over the road map, which is known for any city with all the roads and intersections well-identified. Hence there is no uncertainty at the higher hierarchy. The algorithm can also detect road blockages which lead the vehicle to re-plan, again using Dijkstra's algorithm. As a result, the coarser path always points to the valid path at the coarser level being tracked by the vehicle. The next level of planning consists of Bezier-curve planning, which is optimized by a genetic algorithm (section 2.2). The genetic algorithm uses a set of genetic operators (section 2.3) to compute the optimal path measured by the fitness function (section 2.4). This level deals with uncertainties associated with the relative positions and speeds of the other vehicles and road boundaries. The output of the planner at any time step is the Bezier curve or a trajectory, which is used to move the vehicle. The source of the finer level planning is always the current vehicle position, while the goal is the next distant crossing in the coarser path. As the vehicle approaches its current goal, so the goal is updated to the next crossing. In this manner the coarser path constantly guides the finer level planning by making it reach one crossing after another up to the point where the vehicle's final destination is reached.

Planning is performed independently for all vehicles. Coordination dictates the manner in which the vehicles avoid each other, while each vehicle is planned using finer level planning. The developed coordination strategy (section 3) states that a vehicle only considers the vehicles ahead in its planning which is in consensus with natural human driving wherein one largely considers the prospective motions of the vehicles in front in order to make one's navigation plan. The vehicle trajectories are represented as a space-time graph. While planning, a vehicle to the rear treats all vehicles ahead as dynamic obstacles whose dynamics are known. Every vehicle attempts to move by its highest possible speed (section 3.1).

The resultant coordination strategy cannot display cooperative overtaking, which is handled by plugging in an additional measure in the coordination strategy, according to which a vehicle may ask another vehicle to make some initial turn or alter its speed to better suite its plan. The initial turn is handled by the genetic algorithm while speed is handled by the speed control module. Two heuristic strategies are designed that make use of this measure, namely overtaking (section 3.2) and vehicle following (section 3.3). Both these strategies assess the current scenario and heuristically compute a speed and steering assignment for the different vehicles. The general traversal in the absence of a vehicle ahead is by the path following strategy. The complete approach is discussed in section 2. Section 3 discusses the coordination aspects of the algorithm. Section 4 presents the results of the algorithm. Some discussions are given in section 5 and concluding remarks are to be found in section 6.

The contributions of the paper are (a) The design of a GA which gives results within low computational times for traffic scenarios. (b) Employment of the developed GA for constant path adaptation to overcome actuation uncertainties. The GA assesses the current scenario and takes the best measures for rapid trajectory generation.

(c) The use of traffic rules as heuristics to coordinate between vehicles. (d) The use of heuristics for constant adaptation of the plan to favour overtaking, once initiated, but to cancel it whenever infeasible. (e) The approach is tested for a number of diverse behaviours including obstacle avoidance, blockage, overtaking and vehicle following.

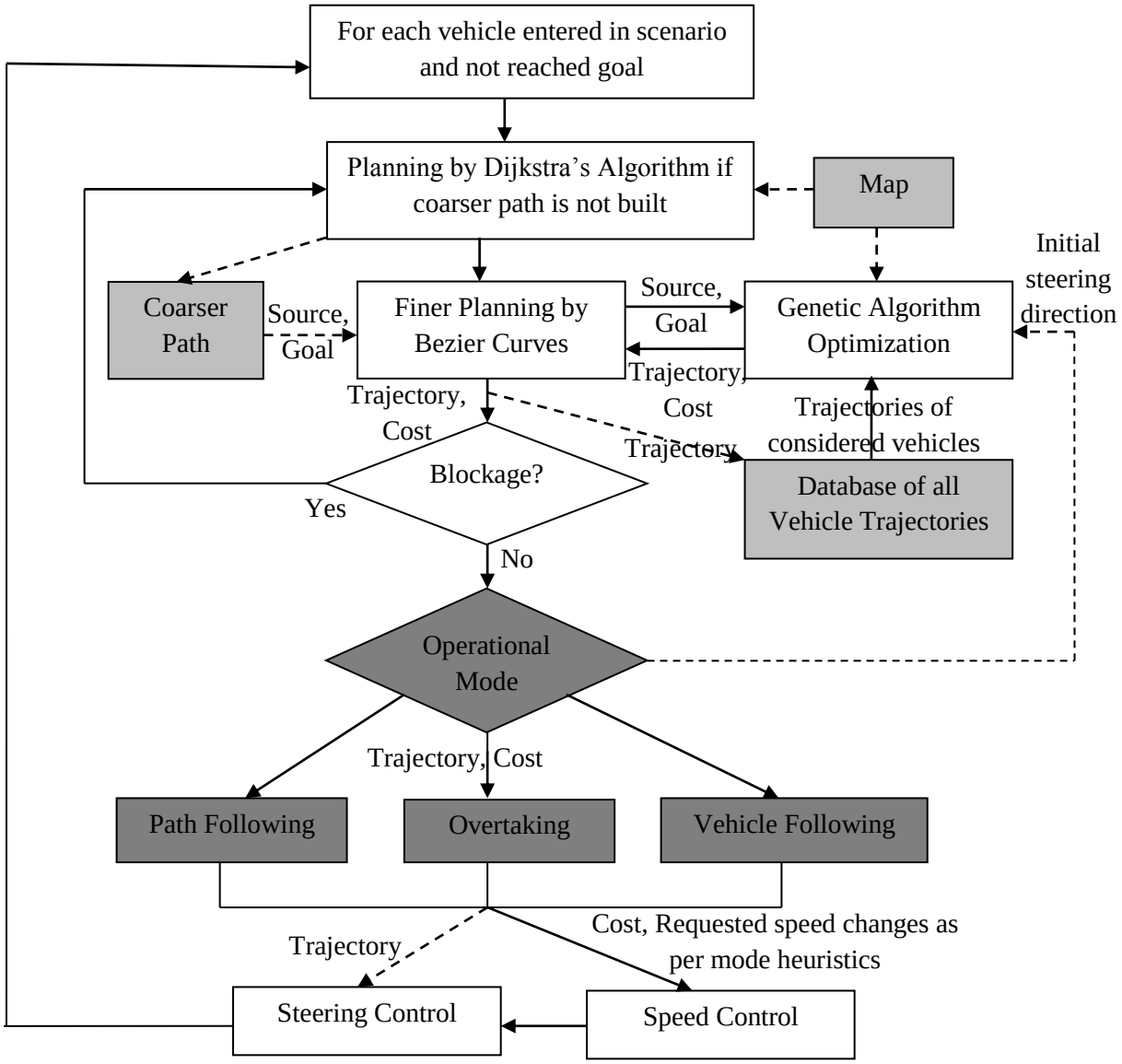


Figure 1: General planning algorithm.

2. The Algorithm

The problem considered here consists of motion planning of N robotic vehicles. Each robotic vehicle R_i has a predefined source S_i and a predefined goal G_i. It is further assumed that the specification of all the roads, in terms of their start, end and connectivity with other roads, in the geographical map is known apriori. The planning must ensure that each vehicle reaches its destination, and that there is no collision with any other vehicle along the way. The road may have static obstacles which can all be sensed as and when they appear.

For simplicity we assume that all vehicles are rectangular in shape with length l_i and width w_i. Every vehicle starts from its source S_i at a time T_i and reaches its goal G_i. The vehicle is assumed to disappear after its journey has been completed. At any time t during its journey, the vehicle is at a position X_i (x_i, y_i) measured in terms of global coordinates, moving with a speed of v_i in a direction of θ_i.

Each vehicle cannot be allowed to approach too closely to an obstacle (which may be another vehicle). This allows scope for measurement error as well as enabling a vehicle to make an emergence stop, if absolutely necessary, to avoid a collision. Hence each vehicle must always keep a clear minimum distance of d_1 from its front and d_2 from its side. Here the minimum front distance d_1 and side distance d_2 are given by equations (1) and (2)

$$d_1 = c_1 v_i^2 \quad (1)$$
$$d_2 = c_2 v_i^2 \quad (2)$$

where c_1 and c_2 are constants.

2.1 Dijkstra's Algorithm

The topmost level planning is provided by Dijkstra's algorithm, which is a single source algorithm that computes the shortest path between any source and any target in the input graph [37]. For any vehicle R_i the input comprises of its source S_i, goal G_i and the road map graph. The road map is well-known for all cities and this can be directly fed to the algorithm. The source S_i and goal G_i of any vehicle is added as vertex before computing its path. The weight of any edge is given by the length of the physical path it represents. The map is shown in Figure 2. For any robotic vehicle R_i, the output is a set of crossings (or vertices) that it must traverse in strict order. Hence the path of R_i becomes $S_i \rightarrow P_i^1 \rightarrow P_i^2 \rightarrow P_i^3 \rightarrow \ldots \rightarrow P_i^v \rightarrow G_i$, as shown in Figure 2. The optimality of Dijkstra's algorithm is based on the assumption that different roads have similar traffic. The road network consists of a large number of vehicles in all, and jointly analyzing all of them for the overall optimal plan is not possible in real time due to the amount of necessary computation.

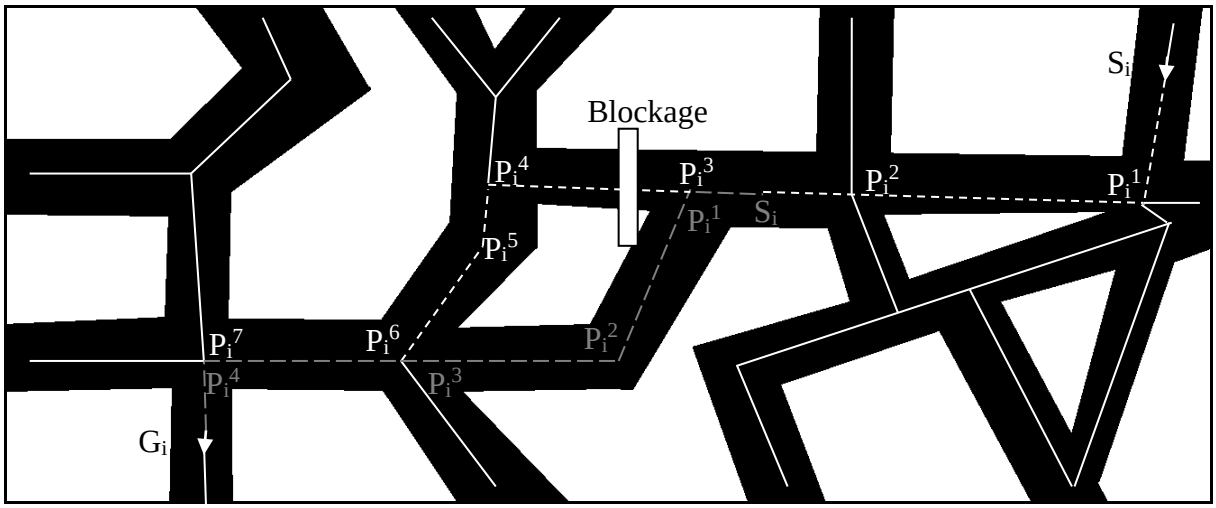


Figure 2: Road Map with Blockage. Each point of intersection of two lines (of paths) is a vertex. The shortest path from source to goal is shown as a dotted line. The list of vertices is returned by Dijkstra's algorithm (S_i, P_i^1 to P_i^7, G_i) shown in white. The path may change upon detection of a blockage. The new path is shown in grey.

As the environment is dynamic, the computed path may, at any instance of time, be found to be blocked. The algorithm must, in this case, take alternative means to act in response to such blockages. This process consists of blockage detection and re-planning. In our algorithm we assume that if the next level of planning algorithm failed to generate feasible solutions for consecutive *block* iterations then the path is regarded blocked. In such a scenario we add the current position of the vehicle as the new source S_i as a new vertex to the road map, the goal is left unchanged. The corresponding connections and weights are updated, to account for the blockage. Dijkstra's algorithm is called again to return a new path. The movements now take place as per this path, as is shown in Figure 2. In the case that no feasible path is possible, the journey of the vehicle is terminated.

2.2 Bezier based planning with Genetic Optimization

The next (finer) level of planner is carried out by the use of Bezier curves [38]. The purpose of the planning algorithm is to generate a path which the vehicle may follow using its own control mechanism. The generated path needs to be as smooth as possible, so that the vehicle may be controlled maintaining high speeds as per the non-holonomic constraints. The other purpose in the use of Bezier curves is to enable the vehicle make efficient turns in all scenarios of crossing, general road turn, or while overtaking. This is shown in Figure 3.

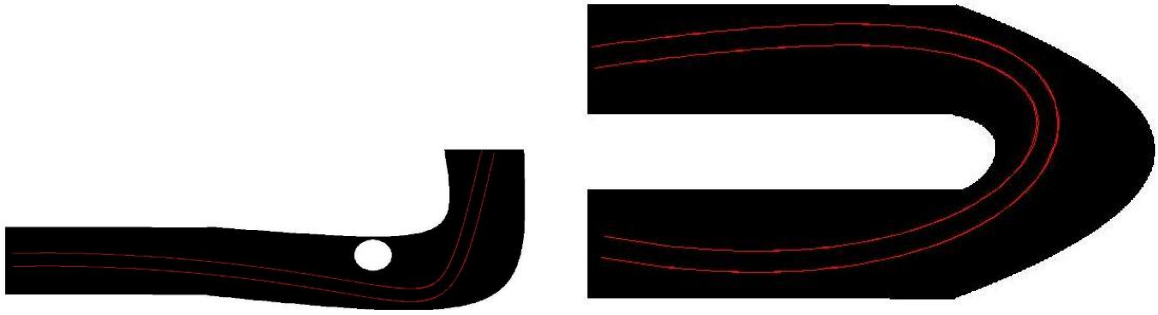


Figure 3: Path generated by the vehicle in multiple scenarios. A vehicle at every instant in time generates feasible paths as per the given map, considering the other moving vehicles (not shown in the figure).

The finer level planning algorithm needs its own source and goal for planning. The source is always the current position of the vehicle (X_i with the vehicle oriented at an angle θ_i). At every time step, the planner generates a trajectory and the vehicle moves as per the same. At the next time step, the changed position becomes the source for the planning. The goal is fixed as the next crossing P_i^j, that is at least η units apart from the current position of the vehicle X_i. As the vehicle continues its journey using this planning, as soon as it comes close enough to the directed goal P_i^j, the goal is changed to the next position in the vehicle's path P_i^{j+1}. However this change is not performed in the case when P_i^j is the final goal [22]. P_i^k always denotes the crossing just behind the vehicle.

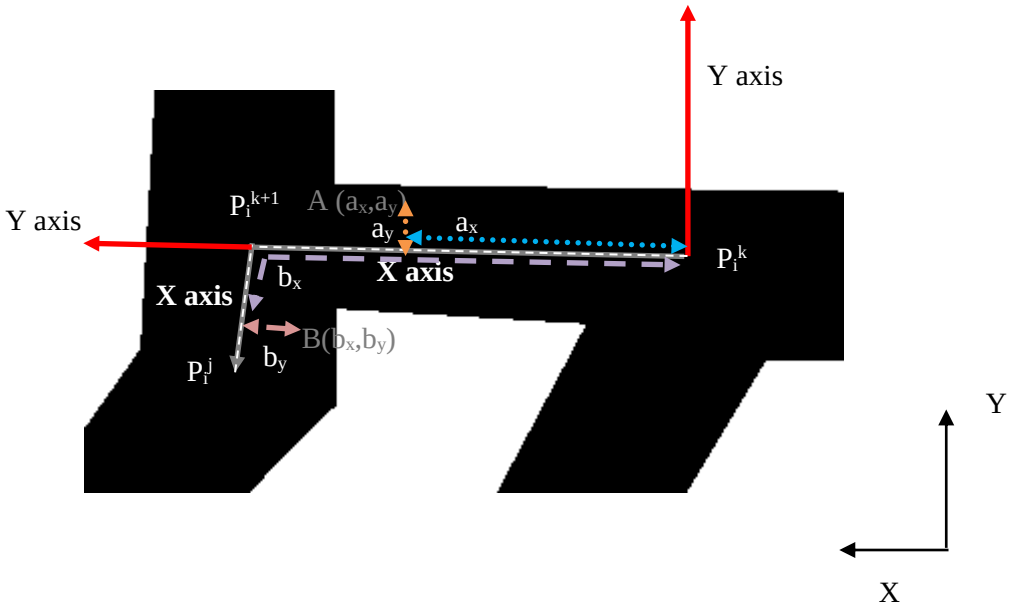


Figure 4: Axis system used for the individual representation in the Genetic Algorithm. The X axis (gray) is the one that joins all the crossings from P_i^k to P_i^j. At any point, the Y axis (red) is perpendicular to the X axis. The representation of two points A(a_x, a_y) (drawn as a dotted line) and B(b_x, b_y) (drawn as a dashed line) as per this coordinate system is shown. a_x (blue) is the distance from P_i^j to this point, measured along the $P_i^k - P_i^{k+1}$ line. a_y (orange) is measured perpendicular to this line. Similarly b_x (purple, tilted line distance) is the distance $\| P_i^k - P_i^{k+1} \|$ plus the distance of the point measured along the $P_i^k - P_i^j$ line. b_y (light red, a negative number) is measured perpendicular to this line.

A Genetic Algorithm (GA) is used to generate an optimal Bezier curve for motion. One of the major tasks in the use of the GA is to devise an individual representation strategy. The Bezier curves may be easily generated by means of control points in the map. For a better solution in the generation of feasible paths, we can take a different two axis representation called as the road axis system. In this system the X-axis is the line joining way-points P_i^k to P_i^j. The Y-axis is perpendicular to the X-axis. The coordinate system and corresponding representation of point is shown in Figure 4.

The complete Bezier curve path specification (or genetic phenotype) consists of (in order) the source X_i, direction maintenance points, a variable number of additional Bezier control points (subjected to a maximum of *MP*), and the goal. The directional maintenance points are added to assure that the Bezier curve generated is initially inclined in the direction the vehicle is currently facing or θ_i. Out of all these points constituting the Bezier curve, only the Bezier control points are non-fixed which are hence the points optimized by the GA. All the control points are placed one after the other to form the genotype. Since different paths may have a different number of control points, the genotype is of variable length. Each point is represented by two numbers representing values across the two axes in the road coordinate system. The bounds for each gene must be known for the optimization process. Every second gene, representing the X-axis component of the control points, can vary from a minimum of 0 to $\sum_{t=k}^{j-1} \left\| P_i^{t+1} - P_i^t \right\|$. Every other gene, representing the Y-axis component of the control points can vary from –w to w, where w represents a measurement which is just larger than the width of the road. This ensures that the GA produces points inside the road. The mapping between the genotype and the phenotype is given by Figure 5.

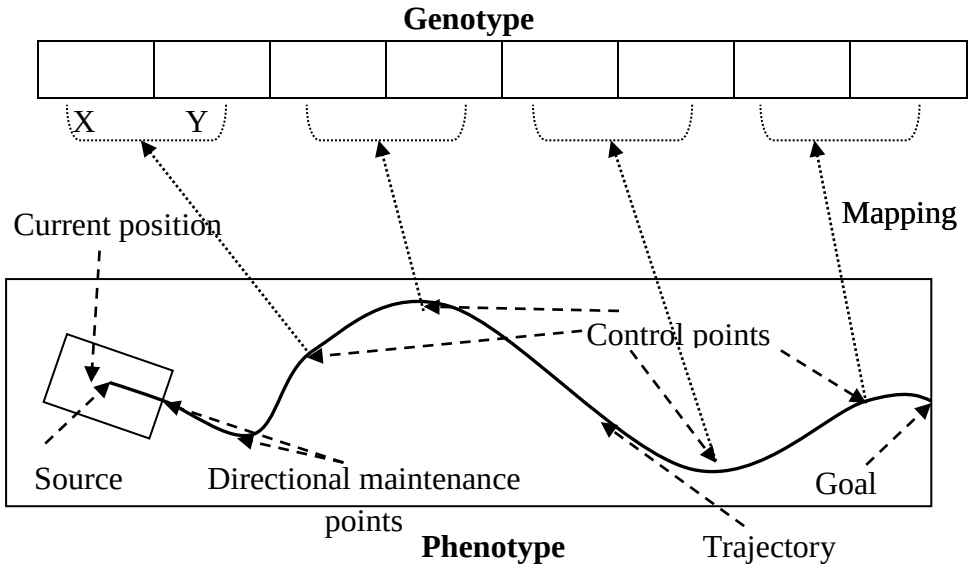


Figure 5: Individual representation strategy. Directional maintenance points generate a curve in the heading direction of the vehicle. Bezier control points (variable in number) constitute the genetic genotype, to be optimized by the genetic algorithm.

We assume here that all the points are sorted as per the values corresponding to the X-axis. This is based on the fact that a vehicle always goes ahead in the direction of the road. Although it may steer by small or large amounts, it never steers large enough to face backward on the road. Hence in a valid curve, a latter point always lies ahead in the road from an earlier point. Further we assume that no point lies behind the current position of the vehicle X_i, as per the road coordinate system. This is based on the fact that the vehicle may have crossed the points behind and these are not needed in the curve since the vehicle would not turn back to touch these points.

The road coordinate axis system used for the generation of individuals enables us to establish points that lie within road boundaries, unlike the Cartesian coordinate axis system where the road may be a portion of the entire map, and hence only some of the points sampled are within road boundaries. In other words the probability of generation of an obstacle-free point is much higher in the road coordinate axis system. Further, as a result of sorting, any path (in the road coordinate axis system) consists of small path segments between any two points which increase the probability of it being feasible; as compared to a randomly produced path in the

Cartesian coordinate axis system wherein two adjacent points would be fairly far apart. By sorting, a number of genotypic representations correspond to a single phenotypic path. This makes the work of the GA easier. A higher probability of generation of a feasible path means the algorithm quickly comes up with a feasible plan which may be optimized in later iterations.

2.3 Genetic Operators

Evolution in the GA is performed by a variety of genetic operators. The first operator used is 'repair'. This operator ensures that the points are sorted as per the values of the X-axis in road coordinate system, all points lie ahead of the current position of the vehicle, and that the individual being considered contains the smallest number of control points. For sorting, the genotype is converted into an array of points consisting of X and Y coordinates. The array is sorted as per the X component. All points in the array behind the current position are deleted, which signify the points that the vehicle has crossed and are no longer needed in the curve representation. The control points are then deleted one by one, until no further deletion results in a better path in terms of its fitness value. The deletion of points from the main path of the vehicle may well yield a better and shorter path [23]. The operation may be time consuming for most newly generated or random individuals, however as the algorithm proceeds, with minor or no change in the scenario, the optimized individuals are already short and hence need not result in much consumed time. The operator is shown in Figure 6(a).

Every deletion of a control point from the path makes the resultant path shorter and smoother. The initial set of points may contain unnecessary twists and turns. However after this smoothing operation it is ensured that every turn results in the vehicle avoiding an obstacle or remaining within a road boundary. Hence, given a good location of control points, the resultant paths are locally shortest and smoothest. It is assumed that the map does not contain a range of closely packed obstacles such that a generated smoothened path might not be navigable physically by the vehicle. In cases where many objects occur the vehicle may initially fail to generate a feasible path, reducing its speed as a result until it is possible to do so.

The second operator used is 'insert', where random individuals of the highest possible length are generated. These individuals obtain a desirable shape after the application of the repair operator. It is possible that due to scenario changes a trajectory which once looked poor may now be desirable. A converged GA population may not have enough exploratory potential to search for such new and interesting solutions. Hence random individuals are added to the population pool. The next operator used is the standard 'crossover' which generates children by exchanging the characteristics of the parents. Since the number of points (or genes) is variable, the first task is computation of the number of points in the two children [27]. The two children have ceil($(n_1+n_2)/2$) and floor($(n_1+n_2)/2$) points, where n_1 and n_2 are the number of points in the two parents, ceil rounds up to the nearest integer and floor rounds down to the nearest integer. These points are randomly taken from the parents, as per the scattered crossover technique. For each vacant point in the first child, a random decision is made whether to take the point from the first parent or the second parent. The corresponding point from the other parent is given to the second child, if vacant positions are available. Left over points may be used to fill in vacant positions in the children. The operator is shown in Figure 6(b).

The next operator used is 'mutation', which changes the value of some point as per the mutation rate. In other words the operator moves the point by some small magnitude over the road to affect the trajectory. The total percentage change that the operator may make is determined by the mutation rate, which is taken from a Gaussian distribution. The relevant operator is shown in Figure 6(c). The final operator used is 'elite', which transfers the best individuals of one generation to the next generation. This operator ensures that the best solution found so far is not killed by the other operators, but is retained till the end and can be used for moving the vehicle. Stochastic universal sampling is used to select the individuals for the different operators. Rank based scaling is used where the different individuals are assigned weights proportional to their rank in the population pool. The individuals are represented in a weighted roulette wheel with the share of the wheel proportional to its weight. A single turn of the wheel is used to select the required number of individuals. The initial population is generated randomly.

Ideally the optimal individual of a generation is still the optimal individual even after motion of the vehicle as per the travel plan, unless there is a change in P_i^k or P_i^j. This is because the Bezier curve remains the same with the start point shifted to the current position of the vehicle, as the vehicle moves. If the vehicle has crossed a control point it would be naturally deleted by the repair operator. In case the vehicle physically crosses P_i^k or P_i^j (in other words when it physically crosses a crossing or there is a change of goal), there is a change of the coordinate system across the movement. Hence the complete population is re-initialized.

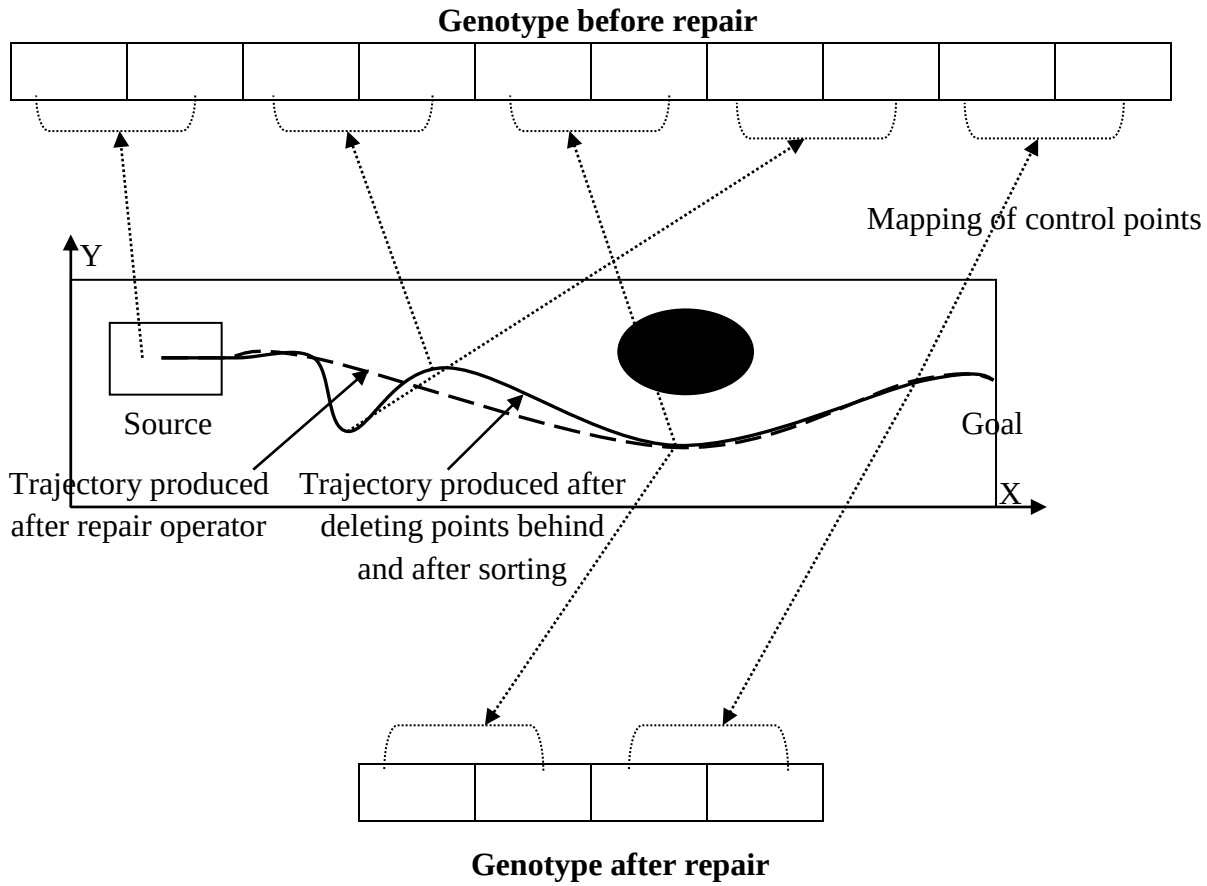


Figure 6 (a) Repair Operator. All points are sorted as per the values of the X-axis, points behind the vehicle are deleted, and control points are deleted till fitness improves.

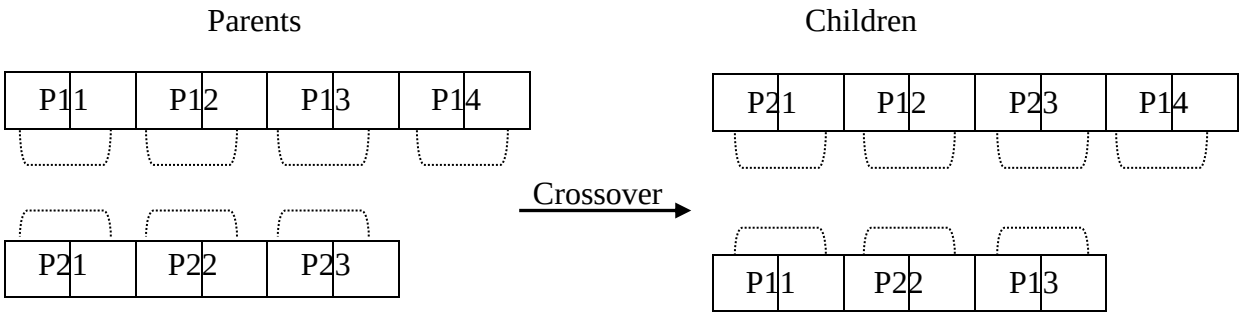


Figure 6(b) Crossover Operator. Since parents are of different length, children are made of mean lengths with random genes exchanged. Symbols denote which gene from parents goes to which child.

2.4 Path Fitness Evaluation

The last part to be considered in the implementation of the GA is the fitness function. We take the fitness function here to be the length of the path. However at the same time a penalty is added for infeasible paths. The penalty is proportional to the length of the segments of the infeasible paths which lie inside obstacles. This creates evolutionary pressures for the GA to minimize the length of the infeasible path up to the point when the infeasible path has been completely eliminated (if possible). The intent is always, of course, to avoid the obstacles, whilst maintaining the minimum safety distance. The resultant overall fitness function may therefore be given by equation (3).

$$Fit = l + \alpha l_1 + \beta l_2 \quad (3)$$

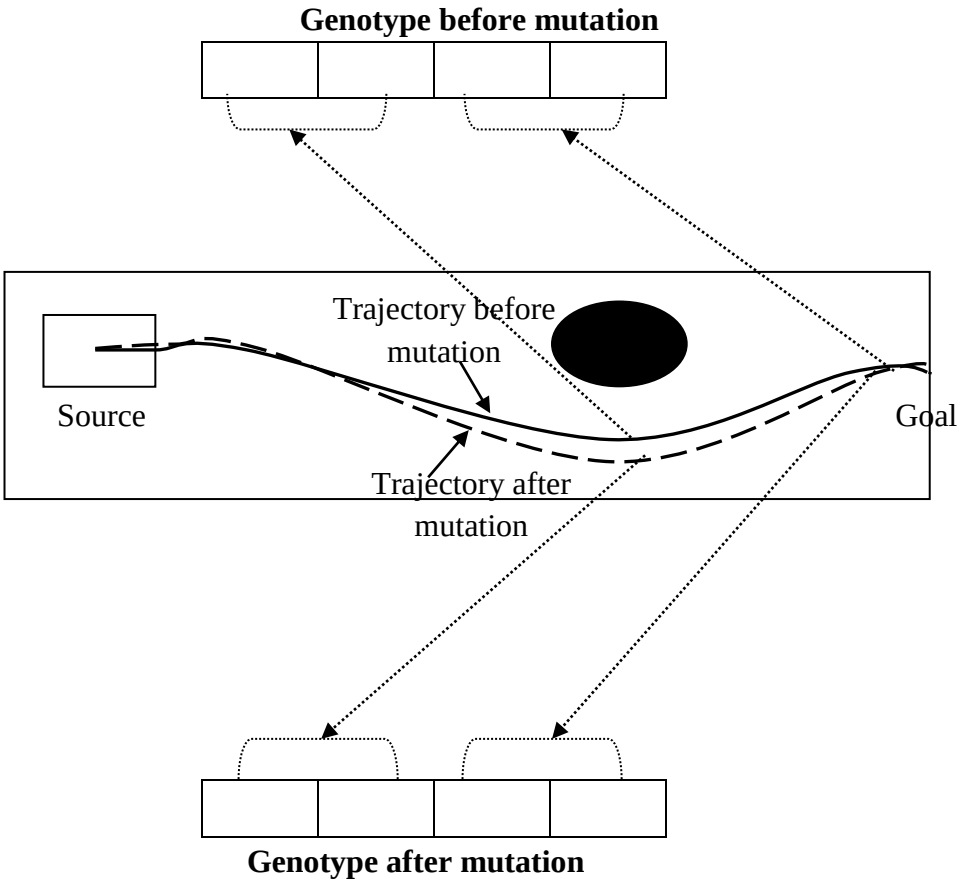


Figure 6(c) Mutation Operator. Points are moved by a maximum magnitude of 'mutation rate' picked from a Gaussian distribution.

Here l is the total length of the Bezier curve, l_1 is the length of an infeasible segment and l_2 is the length of the path segment where the minimum safety distance is violated. $\alpha > \beta > 1$. The GA first tries to reduce the length of the infeasible path. If and once a feasible trajectory is obtained, the GA optimizes to reduce the length of the trajectory without the needed safety distance. Then the focus is to minimize the path length.

It is evident that the fitness function of the GA needs to optimize a number of factors considering the vehicle dynamics such as journey time, vehicle speed, path smoothness, distance from obstacles, sharpest turn, etc. However, unlike planning for mobile robots, roads are generally smooth enough which in itself practically ensures that any general path smoothing strategy leads to smooth curves. On top of this vehicles are restricted to low speed as per the traffic laws in force which means that the allowable speed considering non-holonomic constraints is much higher than the actual speed of travel of the vehicle. For the same reasons there is, in reality, no difference between minimizing time, minimizing path length, or increasing smoothness. The safety distance feasibility measure is similar to the normally encountered factor ensuring a minimum distance from obstacles in mobile robotics.

The factors of safety distance (and feasibility) and path length are clearly contradictory. Shorter paths are possible by violating the safety criterion and keeping as many points in the curve as possible very close to the boundary. Unlike other research in this area we specifically add this factor rather than checking for non-holonomic constraints or smoothness, and allowing the vehicle to go close to the boundary or obstacles. This is done to account for sensor and actuation uncertainties.

Computation of the feasibility and path length, by point to point iteration of the Bezier curve, may be a very costly procedure and the results of the GA might not be realised in real time. Hence the computation is performed at points interleaved by some minimum separation, where the separation decreases along the curve.

This means that in the initial segment of the curve, each point is checked and as we go further still, lesser points are checked.

3. Coordination

In decentralized planning, every vehicle is planned separately. Two vehicles, planned separately, may thus have conflicting trajectories if they do not consider each other in their planning. Both vehicles cannot consider each other's trajectories in their planning, since the trajectory computation of the first vehicle is dependent on the availability of the trajectory of the other vehicle and vice versa. The problem is more complicated with the presence of multiple vehicles, wherein any set of vehicles may have conflicting trajectories, while the alternative trajectories may be further conflicting. Coordination deals with the manner in which such conflicts are handled, or the manner in which vehicles avoid each other, however this only occurs while each vehicle is planned separately in a decentralized manner.

In addition to the static obstacles, the vehicle needs to ensure that it does not collide with any of the other vehicles. This is done by a space-time graph approach [39]. Each vehicle maintains a copy of the current plan being followed in the form of hash map of vehicle positions, hashed with time.

Every vehicle considers only the vehicles that are ahead of it at the current time, although they can be moving in any direction. Hence this is an implementation of a priority based system, where the priorities are founded on the position of the vehicles. This is (once again) inspired from natural driving, where a human driver mainly considers the front view while making their trajectory plan. The purpose of the algorithm is to use the traffic heuristic (of overtaking and vehicle following as is presented in section 3.2 and 3.3) inspired coordination strategy rather than a coordination strategy which jointly optimizes the trajectories of all the vehicles. A traffic inspired heuristic is capable of quickly generating feasible and near-optimal paths.

Coordination dictates the manner in which any two vehicles avoid each other. As per the designed heuristic strategy, a rear vehicle is responsible for avoiding a collision with the vehicle in front. This strategy is however non-cooperative and there is no incentive for a front vehicle to aid a rear vehicle in getting a better path. Hence an additional clause is added that a rear vehicle may request a front vehicle to veer to a particular side, if and by whatsoever amount possible. Whether the request is to be made and if yes in what direction depends upon the overtaking and vehicle following heuristic discussed in section 3.2 and 3.3. The magnitude by which the vehicle in front veers upon being requested denotes the amount of cooperation which is an algorithm parameter. Further, a vehicle to the rear may request a vehicle in front to alter its speed by some amount, which again depends upon the overtaking and vehicle following heuristic.

It is expected that after a short time, paths of vehicles will change due to optimization, emergence of other vehicles, obstacles etc. Hence there is no point in making too far-sighted plans. In natural human driving the focus is on avoiding collisions with nearby vehicles and not avoiding vehicles which are a long way off. So in our system, a vehicle need not consider collisions with vehicles which are recorded after a large span of time. If a vehicle, faced by a possible collision, is already inside the region where the unsafe distance threshold is breached, slowing of the vehicle will take place.

The algorithm employed here uses a form of communication in which one vehicle can communicate with another vehicle and to update its planned path as a result. Initially when a new vehicle comes into the scenario, it is far from any of the other vehicles and hence even if communication takes a reasonable time to be established, it is acceptable. Once communication is established however, each vehicle knows each others' plans and makes necessary corrections so that an overall feasible travel plan is generated. Plans then only need to be re-communicated when there is a large deviation in the path actually travelled by a vehicle and the one it originally communicated at the time of generation of the plan. Hence the algorithm can handle communication errors to some extent.

3.1 Determining Speed

For a single vehicle operating in a static environment, its trajectory can be computed and traced at any desirable speed without affecting the feasibility. However operational speed is important to consider for the case of multiple vehicles. The relative speed of two vehicles determines the coordination or the manner in which the vehicles avoid each other. An optimal coordination plan in the case of multiple robots deals with the optimal planning of the trajectory as well as the optimal planning of speed. The proposed method uses a path-velocity decomposition [40] scheme wherein path components are optimized by the trajectory planner which works with

the assumption that the vehicle would continue to travel with its current constant speed. Considering the needs of the online nature of the algorithm, the velocity component is optimized by a simple mechanism which attempts to assess the scenario and assign the best current speed of the vehicle as the vehicle moves. The attempt is to assign the maximum speed possible to the vehicle, so long as the increased speed does not result in a collision or loss of safety distance.

A simple heuristic rule is used to set the instantaneous speed of the vehicle. At any time the speed of the vehicle may either be increased by a magnitude of δ or decreased by a magnitude of δ [9]. The speed is always subject to a maximum of v^{max}_i which is a property of the vehicle. In the case when the planning routine indicates that the path planned appears to be without any collision and safe distances can be maintained, an attempt is made to increase the vehicle's velocity. However, if the planned trajectory suggests that a collision is likely or that minimum safe distances cannot be achieved, the velocity is decreased. The resultant velocity is hence given by equation (4).

$$v_i(t+\Delta t) = \begin{cases} \min(v_i(t)+\delta, v_i^{\max}) & \textit{if no penalty} \\ \max(v_i(t)-\delta, 0) & \textit{if penalty} \end{cases} \tag{4}$$

Considering that the sampled time Δt and the speed interval δ are small, and keeping the path being traced as fixed, it should be possible for a vehicle to quickly obtain the highest possible speed to trace the path. On reaching the highest speed, any attempt to further increase the speed would be turned down as it would make a penalty free path penalty prone. At the same time since the current path is without penalty, no decrease in speed would take place. Hence for a constant path, the vehicle can obtain and maintain its highest speed possible. In reality, as the vehicle speed changes, the algorithm would also modify the path which may suite the altered speed better. For a unit iteration there is a small change in speed which causes a small change in path. Hence as the vehicle moves, altering speed and path adjustments would be applied leading to convergence which gives a near-optimal plan for navigation.

Having a collision-free near-optimal navigation plan should normally mean that the vehicles stick to it and navigate it using their control mechanisms. However uncertainties need to be handled. The GA handles uncertainties by maintaining enough separation for the vehicle, at all times, from obstacles, other vehicles and road boundaries. However actuation errors or uncertainties may make the vehicle lie at a position somewhat different from that expected. Such errors can grow with time. These sensing errors may however be corrected with time, as the vehicle approaches obstacles or other vehicles. Hence the path and speed are constantly adapted to cope with these uncertainties, to make the overall path near-optimal and collision free. The experimented results provided are in fact obtained using a lazy control mechanism which is intentionally used to produce large errors. An important reason for using the GA was in fact to overcome these errors. The decision to overtake a vehicle or to follow it instead can be altered based on these uncertainties, which is done outside the GA by a separate decision making module.

3.2 Overtaking

A slower vehicle lying ahead of a faster vehicle, occupying some lateral coverage of the road in use by the faster vehicle behind, is too restrictive for the faster vehicle forcing it to drive at a lower speed to avoid an accident. Overtaking deals with first making the two vehicles lie on different lateral coverage of the road such that the slower vehicle does not block the faster vehicle. It would then be expected that the faster vehicle soon lies completely ahead of the slower vehicle, post which the two vehicles may occupy more favourable lateral positions on the road.

One of the major issues associated with the algorithm is to enable faster vehicles to overtake slower vehicles. As per natural human driving, whenever feasible, a faster vehicle may well overtake a slower vehicle. This makes the travel plan of the faster vehicle more efficient. This section (and section 3.3) gives details on the heuristics which lead to the coordination between vehicles. In the algorithm these heuristics are in the form of a vehicle requesting another vehicle to turn, or in the form of a vehicle requesting the other vehicle to slow down. The heuristics are completely derived on the basis of generally observable driving in the absence of speed lanes for comfortable to risky scenarios. The heuristic is based on the philosophy that vehicles need to cooperate, as far as possible, to allow any possible overtake. A key contribution here is that not only are the plans monitored and constantly adapted to enable the completion of a successful overtake, the decision to overtake itself is monitored and can be altered on sensing a potential threat and re-initiated later.

Initially the faster vehicle is situated behind the slower vehicle when the overtake procedure starts. The overtake procedure is different for the case when no other vehicle (apart from the vehicle overtaking and the vehicle being overtaken) is present, as opposed to the case when there is an additional vehicle in the overtaking zone.

The first case involves an attempt to overtake by a faster vehicle R_1 which is initially behind a slower vehicle R_2 ($v_2^{max} < v_1^{max}$). The first task associated with the overtaking procedure is to move the vehicle R_2 leftward (assuming a driving on the left with overtaking on the right traffic rule) if necessary, in the case when it is in front of R_1. This is important as R_2 does not consider R_1 in its trajectory planning, as R_1 initially lies at the rear of R_2. This move is issued as a request broadcast from R_1 to R_2. As an analogy, the process is similar to blowing a horn, indicating an attempt to overtake. The vehicle R_2 considers the request and attempts to move to the left only in the case that the generated path for it to do so is feasible, taking into account possible collisions and minimum safety distances. The opposite rightward motion of the vehicle R_1 is as per the computation of its trajectory. As long as R_2 moves leftward, there is more scope for R_1 to overtake, and its trajectory keeps improving with time. The leftward indicative movements are appended as extra control points in the direction maintenance points in the Bezier based planning. Eventually R_1 and R_2 are reasonably far apart when abreast of each other. In a very short time R_1 is able to evolve a trajectory, considering the trajectory of R_2, such that no collision occurs as well as the overtaking procedure taking place.

Ideally the two vehicles travel as per their own planned trajectories, and the overtaking procedure is completed successfully. However, in this course of time, there may be uncertainties in vehicle control, making the planned trajectories prone to collision. Whenever R_2 detects a possible collision or a minimum safety distance is not maintained, it slows down. This gives greater scope for R_1 to proceed with the overtaking. However when R_1 detects the absence of a minimum safety distance, it may not slow down, but instead it asks R_2 to slow down. This ensures that R_1 does not lag behind R_2 in the overtaking process. It may again be noted that slowing of R_2 reduces the safety distance which needs to be maintained, which may again contribute towards making overtaking feasible. Many times whilst overtaking is being carried out, it may be possible that due to large uncertainties, which may be attributed to a poor control mechanism, overtaking is completely infeasible. Hence, while overtaking, if R_1 detects that there is a collision (excluding the safety distances) for a few consecutive time steps, it abandons the overtaking procedure. In such a case it proceeds to slow down.

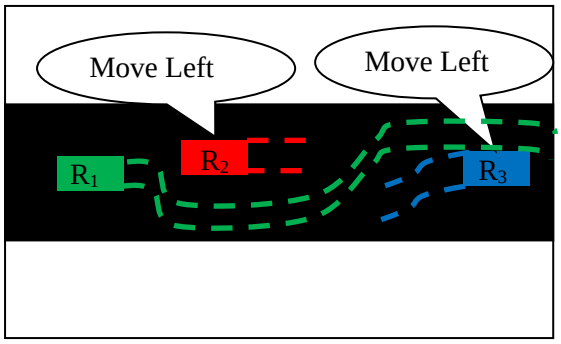
The overtaking mechanism also appears to work well in the scenario of multiple vehicles. However the procedure may involve the cooperation of other vehicles as well, i.e. those not directly involved but which would be affected by the overtaking in terms of disruption to their own trajectory. Here the first task is to decide whether the vehicle should overtake or not. Suppose that the faster vehicle behind is R_1 and needs to overtake the slower vehicle R_2, with the vehicle R_3 ahead and approaching from the opposite direction. Here R_3 may in fact be one or many vehicles of the same kind.

Overtaking, at any time of the journey, is regarded as possible if R_1 can draw a feasible trajectory without colliding with R_2 or R_3 as well as maintaining minimum safe distances from them (assuming that R_2 and R_3 already had collision free trajectories). In such a case, if R_2 or R_3 lie in front of R_1, it would request both of them to move to their left hand side respectively, to allow the overtaking procedure of R_2 to occur. Each of these would however move as requested, only if their generated paths are feasible. The motion of R_1 would then be guided by the planned trajectory of the Bezier curve. As the two vehicles order themselves at their respective left sides, the path of R_1 starts to get both shorter and easier.

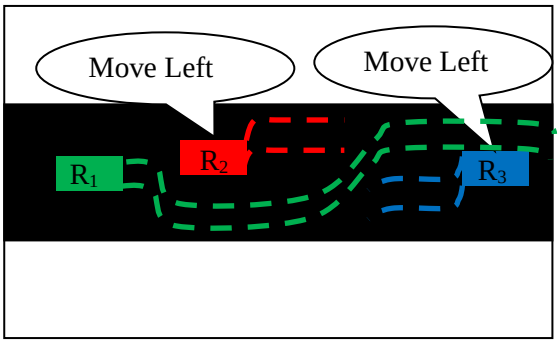
As was the case for two vehicles though, it may not ultimately be possible for all three vehicles to navigate as per their respective individual plans and complete the overtaking procedure without a collision. In such a case a number of actions are required if the overtaking procedure is to go ahead safely. Firstly vehicles R_2 and R_3 need to slow down when the possible collision is detected. This gives ample time for R_1 to comfortably overtake. Vehicle R_1, on seeing a safety distance not being maintained, may ask R_2 or R_3 to slow down further, dependant on which vehicle it is likely to collide with. However vehicle R_1 may itself also need to slow down if a collision is detected. If R_2 or R_3 do not cooperate and take plans contrary to the ones desired to safely complete an overtaking procedure, R_1 would soon come into a collision prone situation and would need to abandon the overtake. The complete overtaking mechanism is shown in Figure 7.

From an analytical perspective, overtaking starts after the vehicles have found a unanimous travel plan by which no collision occurs with wide enough separations during overtaking. The larger the separation, so the more likely it is that overtaking can be actually carried out. Overtaking can be cancelled at any time if the vehicle has a separation which is not large enough for it to fit in. Once cancelled the re-initiation of an overtaking procedure may take place at any time when a feasible plan has been constructed. If a vehicle is moved such that

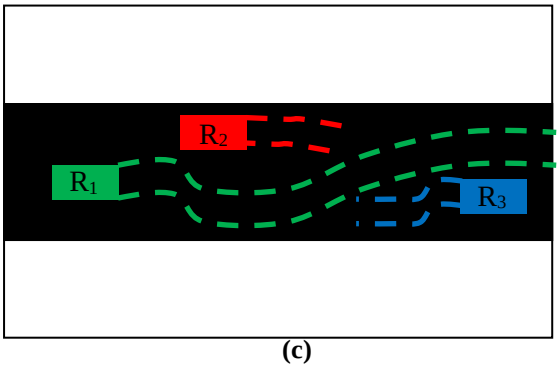
sufficiently wide separation is available at all times, overtaking may go ahead with the same speed profiles of vehicles. However when a vehicle moves, if its separation is not wide enough for it, an attempt is made to increase the separation to the desired threshold by modifying the speed profiles of vehicles. Slowing other vehicles down is usually the best alternative.



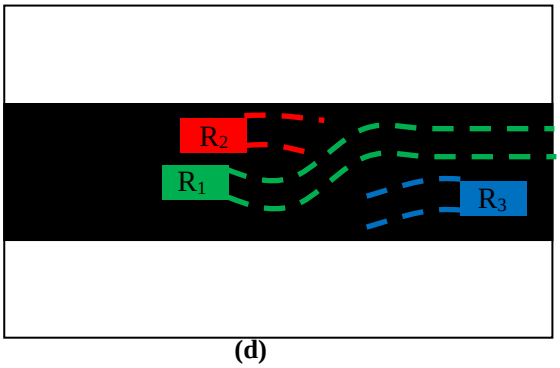
(a)



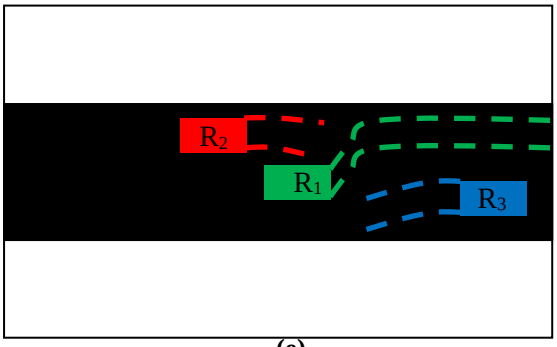
(b)



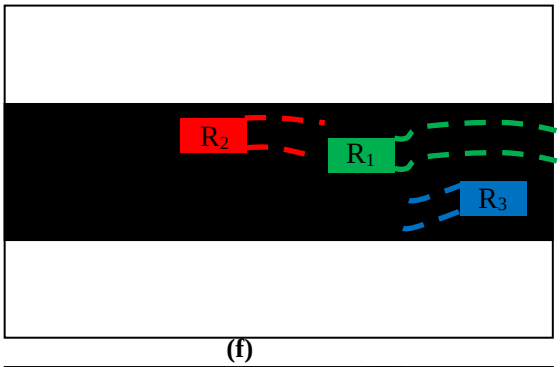
(c)



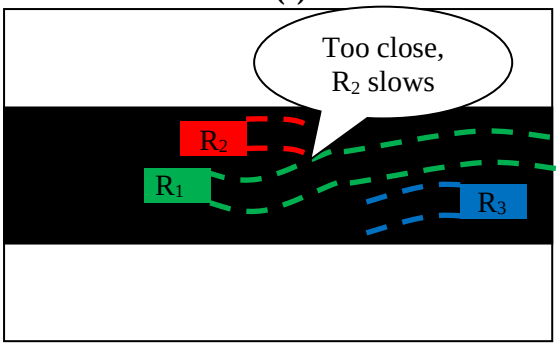
(d)



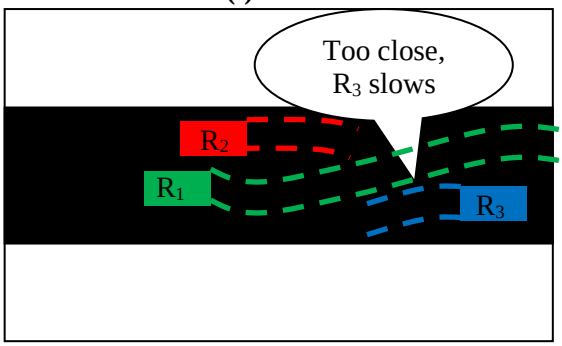
(e)



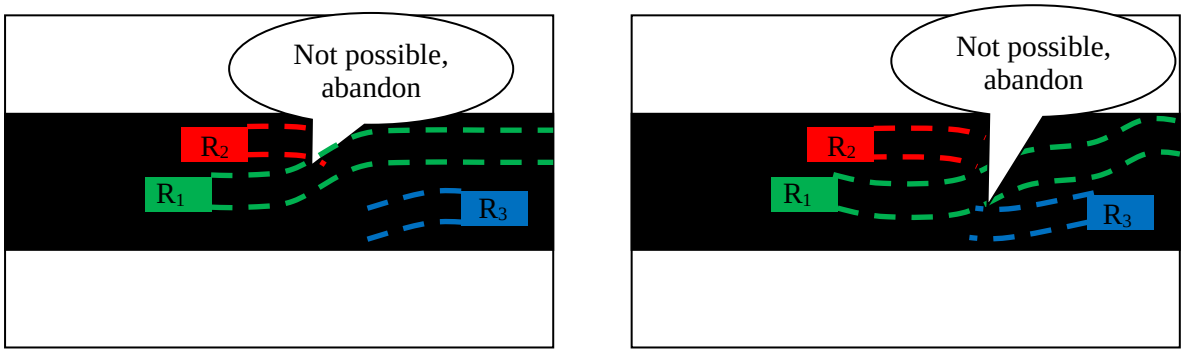
(f)



(g)



(h)



(i) **(j)**

Figure 7: Overtaking with three vehicles. The overtaking mechanism starts when R_1 foresees the possibility of overtaking. This is when it is able to construct a feasible and safe trajectory of its motion such that no collision occurs, whilst considering the feasible trajectories of other vehicles (a). Since R_2 and R_3 are ahead of R_1, these are asked to move in their respective leftward directions, implemented by giving a leftward turn to their Bezier curves (b). R_1 makes a smooth and short trajectory to overtake (c) which is followed by an overtaking manoeuvre (d). In (e) R_1 has completely overtaken R_2 and in (f) it avoids R_3. (g) to (j) show potential problems which can stop the overtaking procedure occurring.

3.3 Vehicle Following

On many occasions, even though a vehicle to the rear of another may be both capable of and desire a higher speed than the vehicle in front of it, overtaking may not be possible. This can be due to a vehicle in the opposite direction approaching an overtaking point which must be avoided during that specific overtaking action. Conversely, in some scenarios the road may not be broad enough to accommodate vehicle overtaking. In such a case the vehicle simply needs to slow down and follow the vehicle ahead. Vehicle following is a common traffic behaviour wherein one vehicle constantly adapts it's speed so as to always maintain a safe separation from the vehicle in front, while the vehicle may prefer to steer so as to be roughly behind the vehicle in front. As a result the two vehicles seem to be following at roughly the same speed with roughly the same lateral positions as long as the vehicle following behaviour is displayed.

Since the vehicle cannot generate any feasible trajectory as per its current speed, it is forced to slow down (i.e. if it doesn't it will collide with the vehicle in front of it). It keeps slowing in this manner till it generates a feasible trajectory. This feasible trajectory may mostly be available when the vehicle is slow enough not to collide with the vehicle in front. This would happen when the vehicle to the rear is travelling at a speed which is equal to or even slower than the vehicle in front. The vehicle may accelerate at times, thereby increasing its speed, however when this makes the distance of separation lower than the safe distance, the vehicle would again need to decelerate. Hence the average speed of the two vehicles would be approximately the same, whilst the scenario remains. If road conditions change then the situation may change as well.

In such a case, even whilst it is still following the vehicle in front, the vehicle to the rear must be as ready as is possible for overtaking. This means a number of things: it must attempt to push the vehicle being followed as far to its left as possible, any oncoming vehicle as far to its left as possible, to position itself somewhere in the middle and to start to accelerate ready for the apriori known change in road scenario. This would enable the vehicle to immediately overtake, as soon as it is possible. This mode of operation is summarized in Figure 8.

4. Results

The entire planning algorithm was simulated in a custom MATLAB design. The complete scenario was read as an image file drawn using a paint utility tool. Scenario specifications required the number of vehicles, their entry times, entry locations, maximum and initial speeds, acceleration limits, etc. All these were read from the scenario specification module. The testing methodology was to individually test each of the features displayed in the algorithm. Accordingly scenarios were set up which were challenging and exploited the capability of the module under test. First the algorithm was tested under a number of scenarios, and then analysis of the vehicle behaviour and the algorithm parameters was carried out, which conveys a greater insight into the algorithm's capabilities.

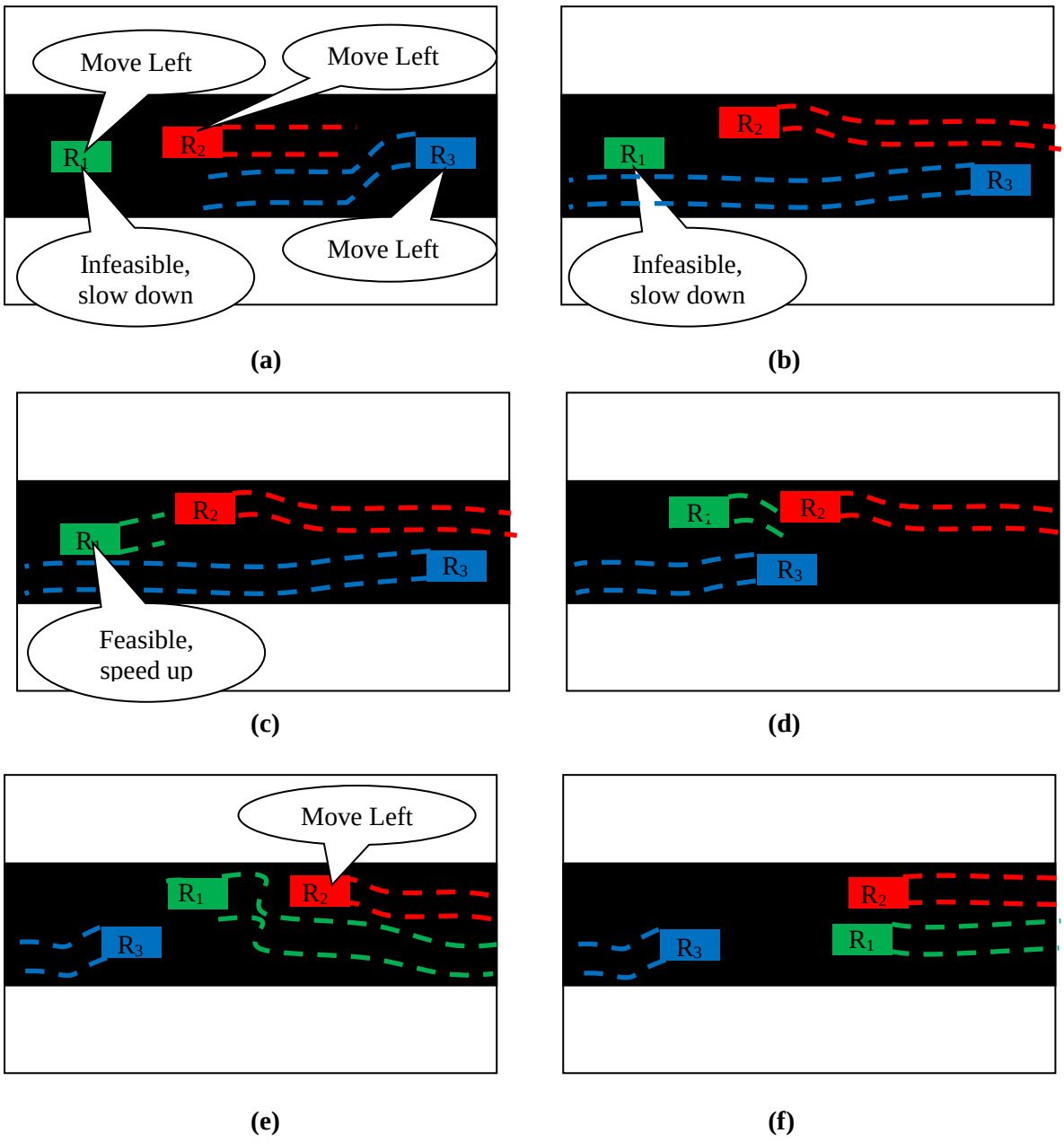


Figure 8: Vehicle following mode of operation. In case vehicle R_1 is unable to compute any feasible trajectory to overtake, it must enter into vehicle following mode. R_2 and R_3 are requested to move leftward as they are directly ahead (a). Vehicle R_1 needs to slow until it is able to compute a feasible trajectory, which would normally mean until it is slow enough to follow R_2 (b). Generation of a feasible trajectory results in an attempt to accelerate (c). R_1 goes through cycles represented in (b) and (c). R_3 is soon about to pass. R_1 and R_2 move as per the planned trajectories (d). Once R_3 has passed, R_1 gets ready to overtake R_2 as discussed in Figure 7. First R_2 is asked to move as far left as possible (e) while R_1 tries to generate feasible and short trajectories. Overtaking is eventually completed (f).

4.1 Higher Level Planning

The first experiment focussed on the higher level planning of the algorithm. While finding a route with a given road map is quite a trivial task, the experiment focussed on the ability of the vehicle to drive on the roads, which largely tested the integration between the higher level planning and lower level planning. To make it challenging for the lower level planning to find safe trajectories as per the higher level path, some obstacles were added on the road. The corresponding path traced by the vehicle is shown in Figure 9(a). This result clearly shows that the

vehicle was able to decide which strategy would be the best in order to avoid an obstacle. The presence of an obstacle though slowed the vehicle's speed a little, which is understandable.

The next feature of the algorithm considered was blockage detection which was tested via another similar experiment. The experiment tested the ability of the lower level planning algorithm to detect the blockage, the higher level planning algorithm to re-plan the path, and the ability to make a smooth transition. The vehicle was successfully able to detect the blockage and re-plan in real time accordingly. A typical traced trajectory is shown in Figure 9(b). In this case there was an alternative path available which was used for navigation. It may be seen that there is a sharp change in the path of the vehicle after the blockage is confirmed by the algorithm. While the algorithm generates infeasible paths, the vehicle's speed is constantly lowered which again helps in making the sharp turns required when a blockage is discovered.

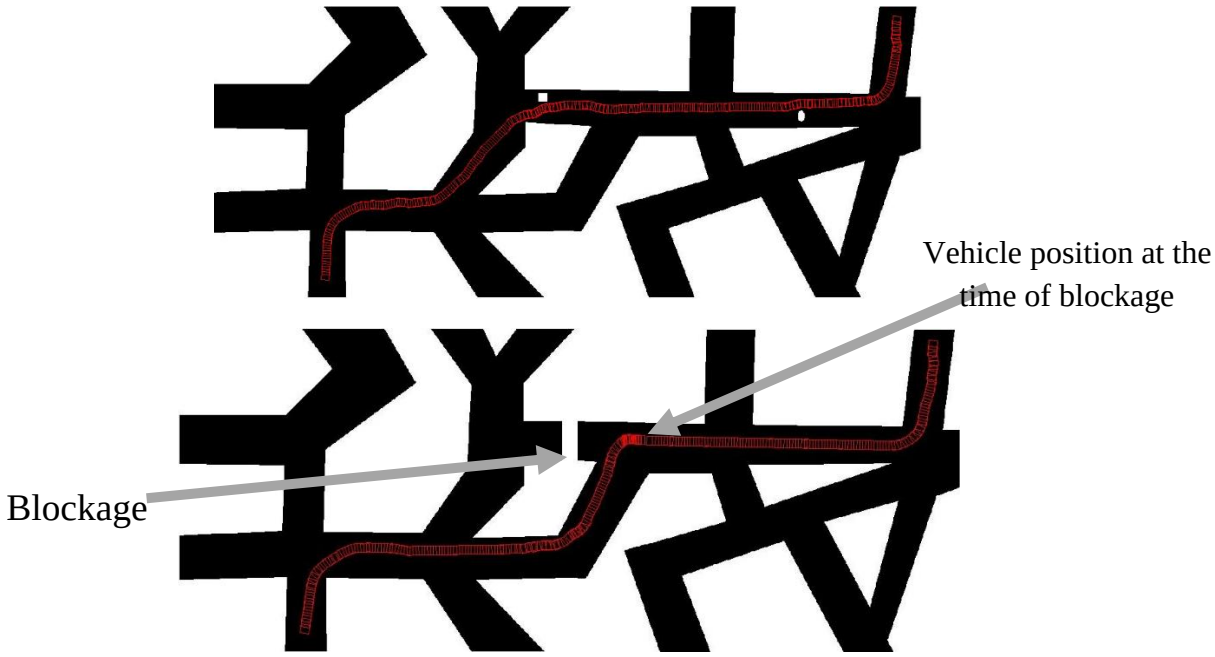


Figure 9: Simulation results over a variety of scenarios. A vehicle is made to move in a variety of maps. (a) tests the ability of the vehicle to steer smoothly around turns and take optimal turns in lesser time, greater speed, and smaller path length. (b) tests the ability of the vehicle to detect blockage and re-plan the path.

4.2 Overtaking

One of the major features of the algorithm is overtaking, which was tested by the next set of experiments. First some simple experiments are presented to highlight the various concepts used in the algorithm. Since overtaking is not done at road crossings, a straight line road was employed as an example indicator. This may be interpreted as a segment of the road in one of the previous scenarios. The first experiment was carried out with two vehicles approaching each other on either side of the road. It was seen that the two vehicles easily coordinated themselves on their left sides. The corresponding trajectory is shown in Figure 10(a). Initially heuristics played a role, in which each vehicle attempted to shift the other vehicle more to its left – this is indicative of the ego type of behaviour exhibited by both vehicles. However the vehicles soon generated feasible trajectories, avoiding each other with a safe distance between.

The second case is one in which a slower vehicle was initially ahead of a faster vehicle. The vehicle to the rear decides to overtake. This case is shown in Figure 10(b) to 10(d). The trajectories follow the norms set at the design and it can be seen that the two vehicles are always separated well apart while the overtaking procedure occurs. The scenario was then reversed with the faster vehicle being ahead of the slower vehicle. In such a case an overtaking action will not take place. This was indeed observed by experimentation. The trajectories of the vehicles are shown in Figure 10(e). Since the vehicle to the rear had a slower speed, computation of its path was not affected by the vehicle ahead. Hence it followed more or less exactly the same path as the vehicle in front of it.

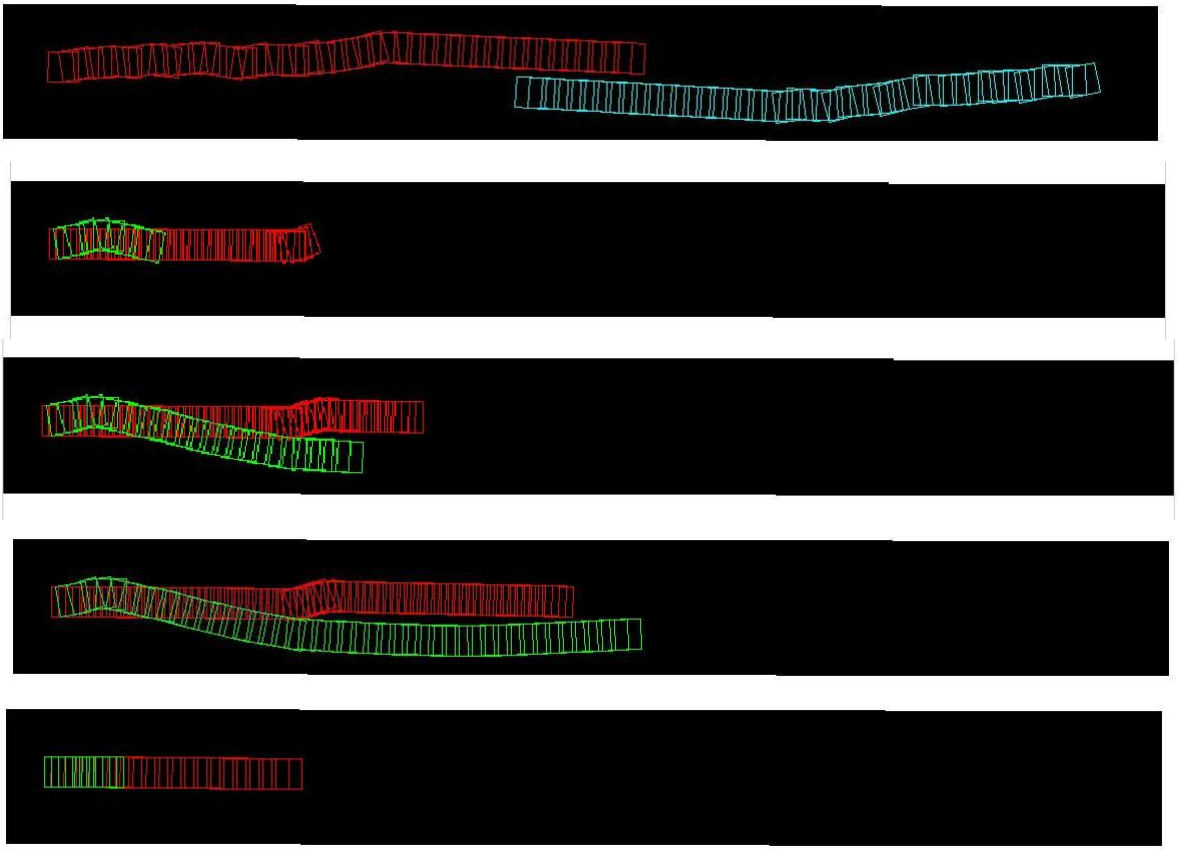


Figure 10: Simulation results for multiple vehicle scenarios. Multiple vehicles are moved along a straight road which (a) tests the ability of two vehicles to avoid collision by moving to the left side of the road. The vehicle is further tested for its overtaking ability. In (b) a higher speed vehicle finds itself behind a slower vehicle. It moves rightward while the vehicle ahead moves leftward. In (c) the two vehicles are fairly well apart and start moving along their computed paths. In (d) overtaking is regarded as complete, as the vehicle is fast and well-ahead. In (e) a slower vehicle is behind a faster vehicle, where both vehicles follow the same path with no collision possible.

Having only 2 vehicles, to some extent, eases the overtaking process. That said, the main ability of the algorithm to be tested is to be able to carry out a close overtaking procedure. This is investigated here by extending the situation to three vehicles, with the third vehicle approaching from the opposite direction. The oncoming vehicle therefore restricts overtaking, which must be completed before any potential collision with the oncoming vehicle. The speed of the oncoming vehicle is kept low enough to make overtaking feasible, but high enough to make overtaking close, which the algorithm must perform. The vehicles have errors associated with their movements, which must be overcome in the small overtaking gap available. The trajectories of this motion are shown in Figure 11(a) to 11(d). It can be seen that the feasibility was correctly computed, based on the time of overtake initiation. The initial overtaking trajectory appears similar to the case with only two vehicles, however the challenge was to timely align the overtaking vehicle to avoid a collision with the oncoming vehicle. In this case a sufficient margin of safety was available after the overtaking procedure.

4.3 Vehicle Following

Vehicle following is a complex behaviour which not only tests the ability of the algorithm to enable one vehicle to follow another vehicle, but also to be prepared for any potential overtaking procedure in the future. Balancing the two acts is a challenge which the algorithm must deal with appropriately. This is tested by an experiment similar to overtaking, with the difference that the speed of the vehicle is adjusted to make an overtaking action infeasible. The trajectories are shown in Figure 12(a) to 12(e). It may be seen that in this case the vehicle first followed the slower vehicle, primarily requiring adjustment (slowing) of speed and location of drive. The vehicle in this case also attempted to be as far to the right as possible. However in case it came close to the oncoming vehicle, it also attempted to drift to the left – balancing the need to overtake with that of avoiding the oncoming vehicle. As soon as the oncoming vehicle had passed, the vehicle in question then proceeded to

overtake the slower vehicle. It can be seen that there was an early (failed) attempt to overtake, but at all times the specified minimum safe distances had to be maintained.

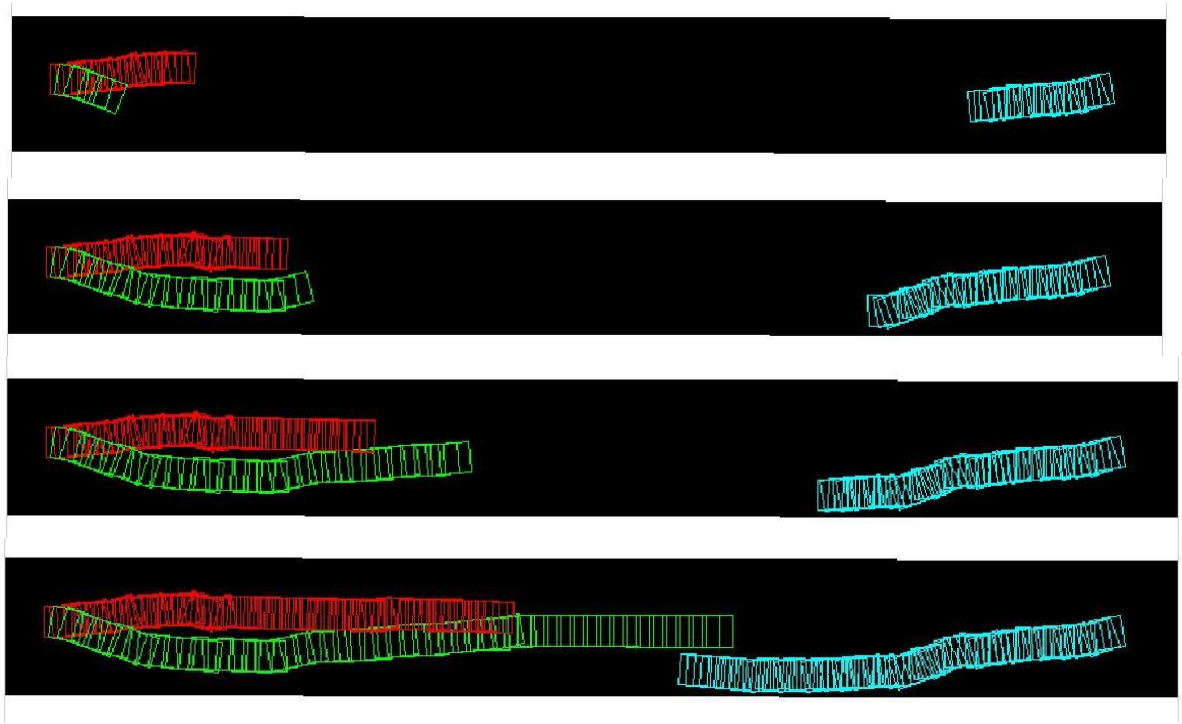


Figure 11: Simulation results for overtaking between three vehicles. A faster vehicle needs to overtake a slower vehicle while another vehicle is heading towards the overtaking zone. Overtaking is computed as feasible by the overtaking vehicle. In (a) the two vehicles attempt to move leftward to give enough room for the overtaking vehicle, which moves rightward. The overtaking vehicle executes an overtaking trajectory in (b). Overtake of vehicle initially ahead is completed in (c), and finally the oncoming vehicle is avoided (d).

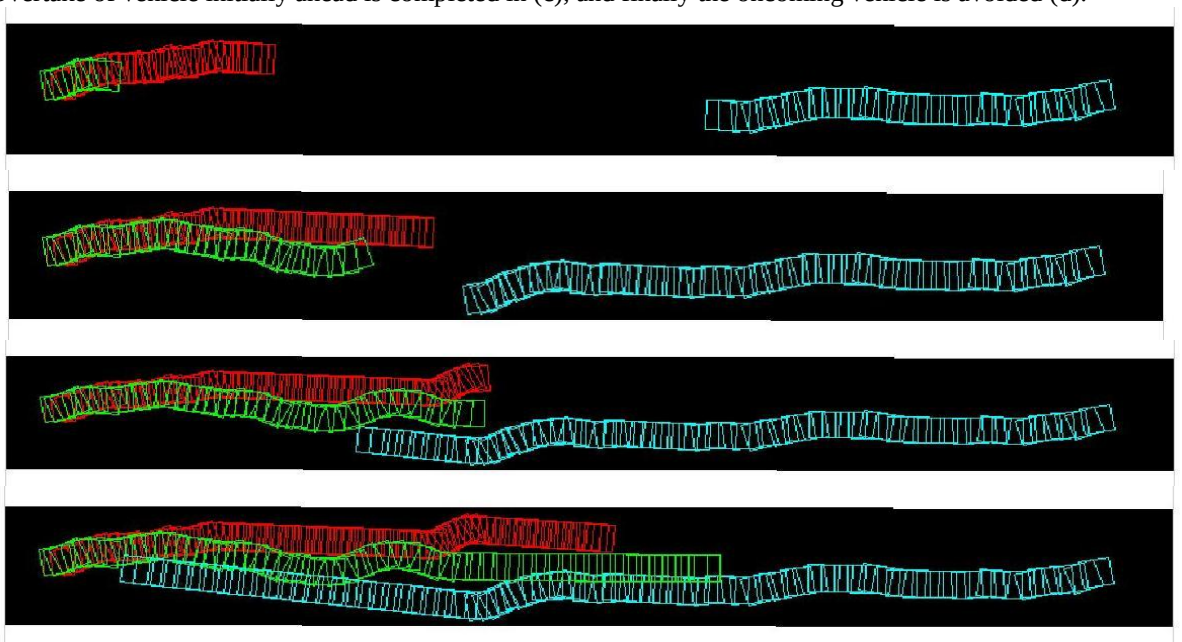


Figure 12: Simulation results for vehicle following behaviour. A faster vehicle behind a slower vehicle is unable to generate a feasible overtaking trajectory due to an oncoming vehicle and hence needs to slow down and follow the vehicle in front. In (a) the vehicle attempts to align itself in the middle of two vehicles, giving enough room for the oncoming vehicle to pass by. The vehicle is expected to drift leftward while the oncoming vehicle does the same to avoid collision in (b). In (c) the oncoming vehicle is completely avoided and overtaking may now take place, which is initiated. The vehicle moves rightward to execute an overtaking trajectory. The other vehicle cooperates by moving leftward. Overtaking is completed in (d).

4.4 Vehicle Behaviour Analysis

Here we metrically analyze the behaviour of the vehicle, when placed in a variety of situations on the road. The intent is to monitor how the vehicle manages its speed and the available road width for its navigation. The factors of efficiency and safety can be contradictory, and yet these need to be balanced for optimal travel. For further discussion, consider that 1 pixel in the developed simulator maps to 10 centimetres on a physical road. This makes the total road length used in our overtaking and vehicle following experiments to be approximately 140 meters. Further consider that a unit time of the simulator is equivalent to 0.1 seconds of physical road travel. Hence a speed of 1 unit distance per unit time in our experiments corresponds to 1 m/s or 3.6 km/h on an actual real world road.

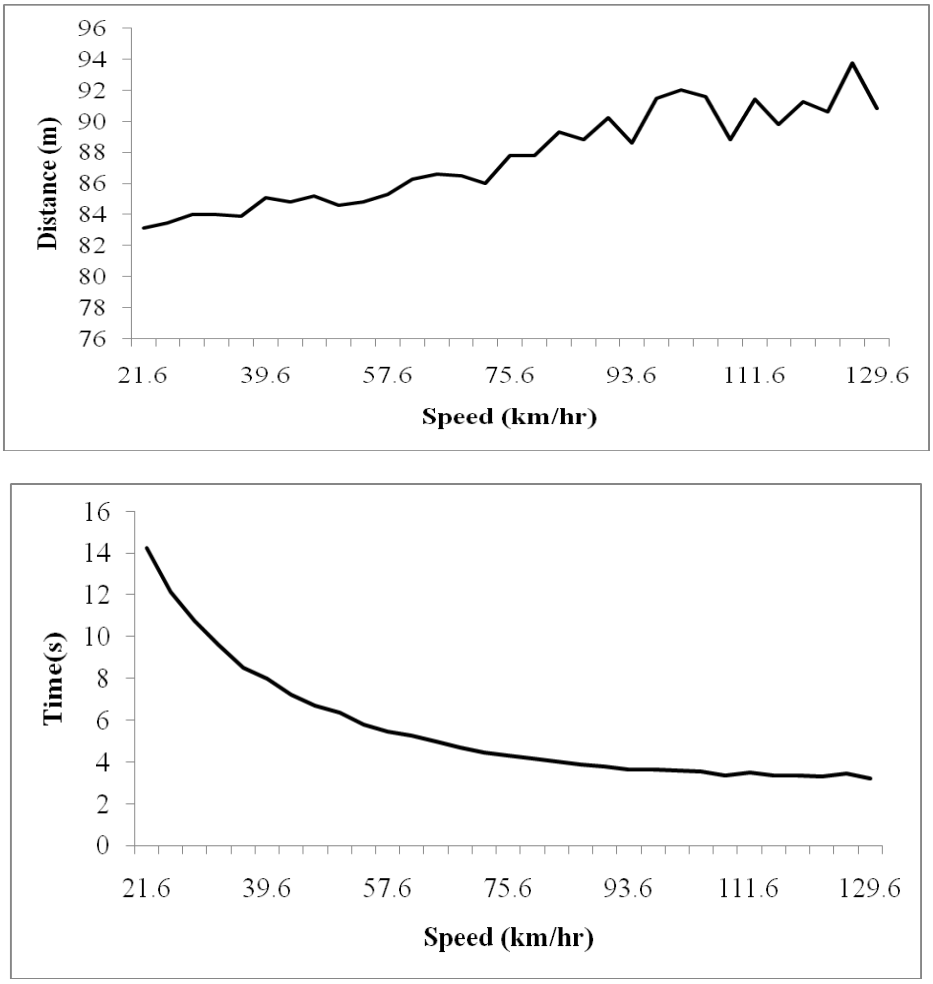


Figure 13: Relationship between the distance and time of travel with speed on a curved road.

Efficiently making turns was one of the prime inspirations behind the use of Bezier curves. We therefore analyzed vehicle performance in this respect. Measurements were taken of the desired trajectory for the vehicle around a curve (in the form of a map), the time of completion of the map and the distance travelled at a variety of maximum speeds. The corresponding plots are given in Figure 13. At lower speeds the vehicle travelled near the inner edge of the curve and hence the distance traversed was low, however this distance increased as the speed increased and the vehicle started travelling towards the outer edge of the curve. Subsequent increases of speed had no effect on distance travelled. This may sound counterintuitive at first, however it must be remembered that at very high speeds the paths generated to safely travel around a curve are infeasible and the vehicle needs to lower its speed until such a path becomes feasible.

In terms of time taken to travel round a curve, results are interesting. Increased speeds usually imply less travel time, however the increased distance travelled round a curve implies a longer travel time. However speed is dominant, the time of travel decreases with increase in speed, although this decrease is not uniform. At higher

speeds the vehicle first speeds down which reduces the average speed (differentiating between average and maximum speed) to some degree.

4.5 Genetic Algorithm Parameter Analysis

The major parameters associated with the algorithm are within the GA which significantly affects overall algorithm performance. Since the solution needs to be returned in real time, it is not possible for the GA either to have a large number of generations for convergence or much time for its optimal solution search. For this reason some GA operators were designed to constantly add diversity to the population pool, whilst others attempted to tune the trajectory and converge. Since both these aspects become important in different situations, the analysis cannot be done on the basis of a set of benchmark situations, where clearly one or other strategy would be of more use. Considering the operation of the repair operator and its ability to quickly produce reasonable paths and small probabilities of invalidation of paths, it can be argued that there needs to be less effort to add diversity and much effort to tune the trajectory. The same strategy is used in the algorithm.

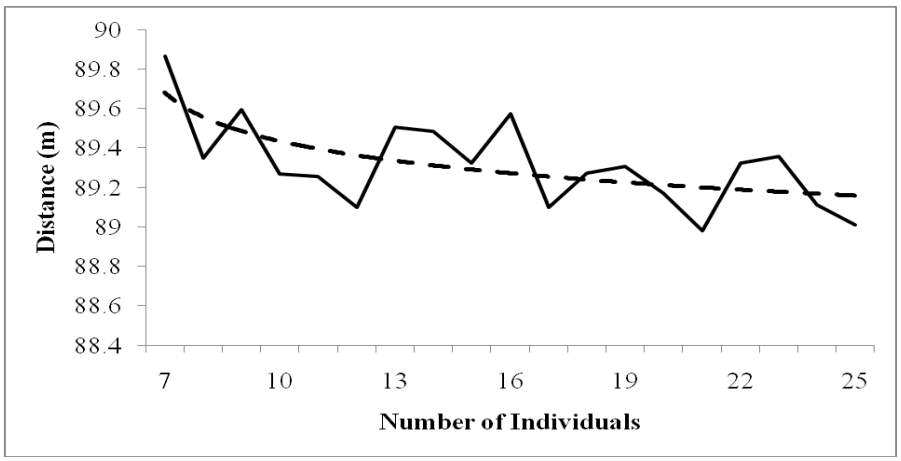


Figure 14: Relationship between distance of travel and the number of random individuals used.

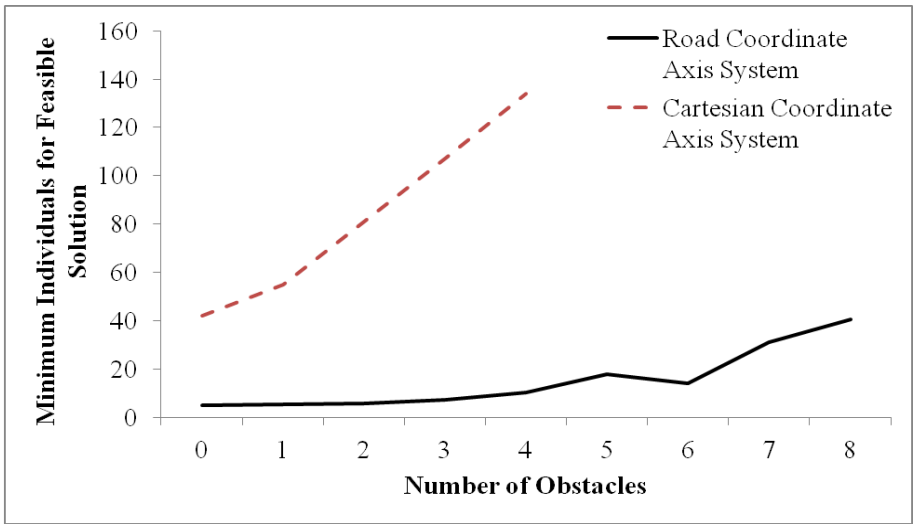


Figure 15: Relationship between the number of individuals needed for generation of feasible path and the number of obstacles in the path.

In such a context, the population size of the GA becomes a major parameter of the algorithm; which needs to be high enough to assure a feasible solution in low number of generations and low enough to assure a small minimum execution time. In all experiments this value was fixed to 20 as a result of the following: At many times the population is reinitialized for the algorithm, and hence it is necessary that the solution generated in a single generation of the algorithm (which makes the immediate move of vehicle) is feasible and near optimal (if

not optimal). Hence we first considered one specific curved scenario and analyzed the number of individuals required to generate a decent feasible path. The total number of individuals or paths generated was increased and the best path length was noted. The corresponding graph is given in Figure 14. It required at least 6 individuals for the generation of feasible paths, avoiding collisions and ensuring minimum safe distances. As the number of individuals was increased, a small decrease in path length was visible. The improvement obtained for a much higher number of individuals was minimal. Experimental evaluations reveal that use of a Cartesian coordinate axis system for the GA individual representation gives infeasible solutions for the part of the graph shown in Figure 14. This gives an indication of the superiority of the road coordinate axis system over the Cartesian coordinate axis system.

Similarly, another experiment was performed with a different number of obstacles on the curved path. The average number of individuals required for the generation of a feasible path was noted. The graph so generated is given in Figure 15. An increase in the number of obstacles means a greater difficulty in the generation of the feasible path as the vehicle needs to make more turns to avoid obstacles. Further, it means fewer possibilities of feasible paths. The increase in difficulty also depends on the place where the obstacle is placed. An increase in obstacles hence increased the number of individuals required, with this increase being more at higher obstacle paths. Any further fitting of obstacles on the path was not possible due to limited road size.

Interestingly, at one instance the number of individuals required actually decreased with an increase in the number of obstacles. This appears to be due to the fact that the effect of one extra obstacle was to guide the Bezier curve in a manner that another obstacle was (automatically) avoided. From both these experiments it can be inferred that a population size of 20 may be a good choice for the simulation as it stands. This ensures the generation of feasible paths for reasonably complex scenarios, whilst the paths are also short enough for simpler scenarios.

In Figure 15 we compared the road coordinate axis system to a Cartesian coordinate axis system, where the points used for individual representation of the GA were initially generated randomly in the road segment being planned. In this experiment we wished to test to what degree the road coordinate axis system was better (or worse) than the Cartesian system. Results clearly indicate a significant improvement in the number of individuals required when using the proposed coordinate axis system. Further for more than 4 obstacles the Cartesian coordinate system was unable to generate a feasible solution even for a very high number of individuals. Hence the Cartesian coordinate system is limited to planning with a reasonably small number of obstacles only.

5. Discussion

Since the algorithm is well-experimented and understood, in this section we carry out analytical comparisons of our work with the different classes of algorithms. The problem is to enable a vehicle to reach its destination, amidst other vehicles. The most important ability of a planning algorithm is to enable close overtaking to take place, in cooperation with the other vehicles. This relies on the ability to generate close overtaking trajectories and formulating a mechanism for the vehicles to cooperate. The algorithms are primarily judged against this ability. The first class we considered here was planning in the presence of speed lanes [30-31]. It has already been argued that speed lanes lead to a lower traffic bandwidth and travelling efficiency in traffic which exhibits a large diversity in speeds and sizes, indeed this is a distinct limitation of such systems. The planning problem in such systems is largely simplified merely to the selection of a lane, which is a discrete choice to make, unlike planning in unorganized traffic where deciding on the vehicle's lateral position from a continuous spectrum is a hard task. In order to judge the close overtaking ability of such systems, consider a vehicle driving on the left of a 2 lane road, with the other lane occupied by a motorbike. It is definitely possible for a motorbike behind the set of vehicles to overtake, which would however not be possible with such systems where the only option is to make lane changes.

More realistic comparisons can be made using algorithms from the domain of mobile robotics, which can be adapted to work in traffic scenarios. Consider, for example, priority based planners [11-12]. In order to test the overtaking capability, consider that a slower vehicle is driving in the centre of a not-so-wide road, with a faster vehicle behind it. Overtaking is possible for the vehicle behind if the slower vehicle in front veers to one side, thereby giving enough space for the faster vehicle to overtake. First consider the slower vehicle is given a higher priority when compared to the faster vehicle, hence planning of the slower vehicle would not consider the presence of the faster vehicle. This means the slower vehicle would continue to travel in the centre of the road, denying the vehicle behind the opportunity of overtaking. The lack of cooperation, in this case, makes overtaking impossible. Now consider the alternative case when the faster vehicle is given a higher priority, as a

result of which it would not consider the slower vehicle in its trajectory planning and would thus travel straight ahead with high speed. The slower vehicle, being planner later, can neither veer left nor right in order to avoid collision with the faster vehicle at the centre of the road. It is further not possible to optimize the priority scheme due to computational constraints.

The search based approaches are not applicable in real time due to the high resolution of the map, for which a convenient method is to use hierarchical planners [22, 27] which break and solve the problem in multiple hierarchies. The initial planning is on lower resolution maps, which is hence fast. The metric is the ability to generate a close overtaking trajectory. In order to do so the width of the road must be precisely known which helps a decision on whether a vehicle can fit in or not. Reducing the map resolution for initial planning would mean that an incorrect or only vague idea of the road width would be available for higher level planning. Regions too close to road boundaries would be regarded as obstacle prone and not used by the algorithm at all. Hierarchical planning in mobile robots mains deals with decision making regarding the selection of wide gaps and hence lower resolution maps and searches are of use. In traffic scenarios, the lateral resolution cannot be reduced which thus affects any overtaking decision making, conversely the longitudinal resolution may be reduced. On top of this, taking far off obstacles (or vehicles) into consideration is clearly not immediately useful as in real time travelling decisions are primarily made based on vehicles and/or obstacles nearby. The approach still needs to be annexed with a cooperative coordination strategy.

Rapidly-exploring Random Trees [17, 18] and Probabilistic Roadmaps [7, 41] have capabilities to quickly sample out the search space to find the optimal trajectory. Such techniques can be used with a coordination strategy as proposed for cooperative overtaking. For close overtaking however, it is important to constantly tune a trajectory up to the point where it is safe and feasible. These techniques place random samples which are only accepted if feasible. Once a sample is placed, its position is never altered or optimized. This can lead to missing a close overtaking opportunity due to poor positions of the samples or lack of effort in tuning the trajectory for feasibility and safety.

Whenever dealing with modular algorithms it is essential to understand the limitations of and issues with the integrated physical devices. Sensors may be noise prone, control systems may not be perfect and communication may be irregular. A strong aspect of our algorithm is how it deals with uncertainty. We do not model the problem in terms of an offline planning phase followed by an online control phase. Rather we model an online planning phase where the plan iteratively adapts to the uncertainties and errors encountered with time. All vehicles constantly adapt themselves to the control errors in each other's motion.

6. Conclusions

Planning and coordination of multiple autonomous vehicles on a road network is an exciting problem that is yet to be deeply studied. From the point of view of a single vehicle, the challenge lies in effective strategic planning at the top level to obstacle avoidance at the lower level. The real time nature of the problem however eliminates the possibility of developing fancy solutions which are time costly. Unlike most mobile robotics studies, the cost of collisions in vehicular systems is likely to be high and must be avoided by all means. The presence of multiple vehicles stresses the need for coordination strategies between vehicles at all times such that no collision occurs between the vehicles and such that each vehicle has a fair path along which to move forward at its own desired speed as much as possible. The practical high diversity in speeds of the vehicles, together with a relatively small workplace make the problem very different from existing (central processing) solutions in the research domain of mobile robotics.

The proposed solution takes some inspiration from traffic rules, which theoretical analysis and experimental observations verify to be optimal, considering a large number of scenarios and the presence of uncertain incidents. These traffic heuristics, used with an evolutionary algorithm for the generation of Bezier curves, are able to navigate the vehicle in a variety of scenarios. The experimented scenarios range from straight roads and roads with a large number of obstacles to close overtaking and infeasible overtaking. This more or less covers the variety of scenarios that the vehicle may face in real life. Uncertainties are the key element that makes software simulations differ from real world experiments. We ensure here the inclusion of uncertainties by making the robot move as per its own kinematic model, and not necessarily the planned trajectory. The planner at times did generate sharp curves, which could not be followed by the vehicle. These errors were corrected with time.

While the experiments worked well, there are a few points of concern that may play a role in use of the algorithm on physical vehicles. Firstly the algorithm assumes that the map is well known, which may not be the

case in all scenarios due to the limitations of sensors and their associated measurement noise. The presence of too many vehicles, traffic jam situations, and sudden emergencies of vehicles extremely close together may further affect the algorithm's performance. These situations are usually also troublesome to humans driving vehicles and only get resolved slowly with time. Much more research is needed to handle scenarios such as mergers, diversions, traffic signals, parking, takeoff from parking and sharp U-turns when operating in the absence of speed lanes. The possibility of having different vehicles networked should point to a nice way of dealing with this problem in the future. Further, a vehicle, while planning, considers only the published results of another vehicle. Actually understanding a vehicle's gestures might reveal more of the true plan. As with natural driving, if a vehicle decides to overtake it must do so with the cooperation of other vehicles. However, ineffective control might make other vehicles slow down to extremely low speeds. The safety distances that humans maintain while driving seem to differ with the situation apparent at the time (not only because of speed). These variations may be seen in a traffic jam situation, overtaking situations, etc. Along with better relation to speed and determination of the speed constant, if this safety distance is effectively modelled to deal with different situations, the overall performance of the algorithm may be boosted.

What we have shown here is a future look at traffic scenarios in which there is a high quality of communication and cooperation between vehicles. Comparing this with present day automotive travel is intriguing. As can be witnessed on roads today, during an overtaking procedure the (slower) vehicle in front usually does little or nothing to enable a (faster) vehicle behind to overtake, the same being true of an oncoming vehicle – indeed a vehicle in front may in fact decrease its speed just to be awkward! On most roads of reasonable size, lanes are employed to retain vehicles within certain rough, safe areas – with the system described here in action, lanes would not be required and many roads could be much narrower whilst still carrying the same volume of traffic. Perhaps the biggest positive in the use of a technique such as that described is that the presently commonplace slowing and queuing at road works could become a thing of the past!

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 Darpa Urban Challenge: Autonomous Vehicles in City Traffic, Springer-Verlag, Berlin Heidelberg, 2009.
[2] S. S. Ge, F. L. Lewis, Autonomous Mobile Robots, Sensing: Control, Decision Making and Applications, CRC Press, Boca Raton, FL, 2006.
[3] 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, J. Field Rob. 25 (2008) 569–597.
[4] T. Arai, J. Ota, Motion Planning of multiple mobile robots, in: Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems, Raleigh, North Carolina, USA, 1992, pp. 1761-1768.
[5] L. E. Parker, Current state of the art in distributed autonomous mobile robotics, Distributed Autonomous Robotic Systems vol. 4, in: L. E. Parker, G. Bekey, and J. Barhen, (Eds.), Tokyo, Springer-Verlag, Japan, 2000, pp. 3–12.
[6] T. Arai, E. Pagello, L. E. Parker, Editorial: Advances in Multi-Robot Systems, IEEE Trans. Rob. Automat. 18(5) (2002) 655-661.
[7] P. Svestka, M. H. Overmars, Coordinated motion planning for multiple car-like robots using probabilistic roadmaps, in: Proceedings of the 1995 IEEE International Conference on Robotics and Automation vol. 2, Nagoya, Japan, 1995, pp.1631-1636.
[8] G. Sánchez-Ante, J.C. Latombe, Using a PRM Planner to Compare Centralized and Decoupled Planning for Multi-Robot Systems, in: Proceedings of the IEEE International Conference on Robotics and Automation, Washington, DC, 2002, pp 2112 – 211.
[9] J. Sewall, J. van den Berg, M. C. Lin, D. Manocha, Virtualized Traffic: Reconstructing Traffic Flows from Discrete Spatio-temporal Data, IEEE Trans. Vis. Comput. Graph. 17(1) (2011) 26-37.
[10] R. Kala, K. Warwick, Planning Autonomous Vehicles in the Absence of Speed Lanes using an Elastic Strip, IEEE Transactions on Intelligent Transportation Systems, 2013, DOI: 10.1109/TITS.2013.2266355.

[11] M. Bennewitz, W. Burgard, S. Thrun, Optimizing schedules for prioritized path planning of multi-robot systems, in: Proceedings of the 2001 IEEE International Conference on Robotics and Automation, Seoul, Korea, 2001, pp 271 – 276.
[12] M. Bennewitz, W. Burgard, S. Thrun, Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots, Rob. Auton. Syst. 41(2-3)(2002) 89–99.
[13] M. A. Potter, The design and analysis of a computational model of cooperative coevolution, PhD thesis, George Mason University, Fairfax, Virginia, 1997.
[14] K.O. Stanley, R. Miikkulainen, Competitive Coevolution through Evolutionary Complexification, J. Artif. Intell. Res. 21(1)(2004) 63–100.
[15] Z. Cai, Z. Peng, Cooperative Coevolutionary Adaptive Genetic Algorithm in Path Planning of Cooperative Multi-Mobile Robot Systems, J. Intell. Rob. Syst. 33(1)(2002) 61-71.
[16] W. Mei, W.Tie-Jun, Cooperative co-evolution based distributed path planning of multiple mobile robots, J. Zhejiang Univ. – Sci. A, 6(7)(2005) 697-706.
[17] Y. Kuwata, S. Karaman, J. Teo, E. Frazzoli, J. P. How, G. Fiore, Real-Time Motion Planning With Applications to Autonomous Urban Driving, IEEE Trans. Control Syst. Technol. 17(5)(2009) 1105-1118.
[18] R. Kala, K. Warwick, Multi-Vehicle Planning using RRT-Connect, J. Behav. Rob. 2(3) (2011) 134-144.
[19] H. Asama, A. Matsumoto, Y. Ishida, Design of an Autonomous and Distributed Robot System: ACTRESS, in: Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems, Tsukuba, Japan, 1989, pp 283-290.
[20] H. Asama, K. Ozaki, H. Itakura, A. Matsumoto, Y. Ishida, I. Endo, Collision avoidance among multiple mobile robots based on rules and communication, in: Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems, Osaka, Japan, 1991, pp 1215-1220.
[21] E. Vendrell, M. Mellado, A. Crespo, Robot planning and re-planning using decomposition, abstraction, deduction, and prediction, Engg. Appl. Artif. Intell. 14(4)(2001) 505–518.
[22] R. Kala, A. Shukla, R. Tiwari, Fusion of probabilistic A* algorithm and fuzzy inference system for robotic path planning, Artif. Intell. Rev. 33(4)(2010) 275-306.
[23] J. Xiao, Z. Michalewicz, L. Zhang, K. Trojanowski, Adaptive evolutionary planner/navigator for mobile robots, IEEE Trans. Evol. Comput. 1(1)(1997) 18-28.
[24] O. Khatib, Real-Time Obstacle Avoidance for Manipulators and Mobile Robots, Proceedings of the 1985 IEEE International Conference on Robotics and Automation , St. Louis, Missouri, 1985, pp. 500-505.
[25] F. Fahimi, C. Nataraj, H. Ashrafiuon, Real-time obstacle avoidance for multiple mobile robots, Robotica, 27(2)(2009) 189-198.
[26] E. K. Xidias, P. N. Azariadis, Mission design for a group of autonomous guided vehicles, Rob. Auton. Syst. 59(1)(2011) 34-43.
[27] R. Kala, A. Shukla, R. Tiwari, Dynamic Environment Robot Path Planning using Hierarchical Evolutionary Algorithms, Cybern. Syst. 41(6)(2010) 435-454.
[28] M. A. P. Garcia, O. Montiel, O. Castillo, R. Sepulveda, P. Melin, Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation, Appl. Soft Comput. 9(3) (2009) 1102–1110.
[29] M. Lepetic, G. Klancar, I. Skrjanc, D. Matko, B. Potocnik, Time optimal path planning considering acceleration limits, Rob. Auton. Syst. 45 (2003) 199–210.
[30] R. Schubert, K. Schulze, G. Wanielik, Situation Assessment for Automatic Lane-Change Manoeuvres, IEEE Trans. Intel. Trasport. Syst., 11(3)(2010) 607-616.
[31] A. Furda, L. Vlacic, Enabling Safe Autonomous Driving in Real-World City Traffic Using Multiple Criteria Decision Making, IEEE Intel. Trasport. Syst. Magz., 3(1)(2011) 4-17.
[32] J. E. Naranjo, C. González, R. García, T. de Pedro, Lane-Change Fuzzy Control in Autonomous Vehicles for the Overtaking Maneuver, 9(3) (2008) 438-450.
[33] H. Jin-ying, P. Hong-xia, Y. Xi-wang, L. Jing-da, Fuzzy Controller Design of Autonomy Overtaking System, in: Proceedings of the 12th IEEE International Conference on Intelligent Engineering Systems, Miami, Florida, 2008, pp 281 – 285.
[34] E. Onieva, J.E. Naranjo, V. Milanés, J. Alonso, R. García, J. Pérez, Automatic lateral control for unmanned vehicles via genetic algorithms, Appl. Soft Comput. 11(1)(2011) 1303–1309.
[35] G. Hegeman, A. Tapani, S. Hoogendoorn, Overtaking assistant assessment using traffic simulation, Transport. Res. Part C 17(6)(2009) 617–630.
[36] F. Wang, M. Yang, R. Yang, Conflict-Probability-Estimation-Based Overtaking for Intelligent Vehicles, IEEE Trans. Intel. Trasport. Syst., 10(2)(2009) 366-370.
[37] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, An Introduction to Algorithms 2nd Edition, MIT Press, Cambridge, MA, 2001.
[38] R. H. Bartels, J. C. Beatty, B. A. Barsky, Bézier Curves. An Introduction to Splines for Use in Computer Graphics and Geometric Modelling, Morgan Kaufmann, San Francisco, CA, 1998, pp. 211-245.

[39] S. Carpin, E. Pagello, An experimental study of distributed robot coordination, Rob. Auton. Syst. 57(2)(2009) 129-133.
[40] K. Kant, S.W. Zucker, Toward Efficient Trajectory Planning: The Path-Velocity Decomposition, Int. J. Rob. Res. 5(3)(1986) 72-89.
[41] L. E. Kavraki, M. N. Kolountzakis, J. C. Latombe, Analysis of probabilistic roadmaps for path planning, IEEE Trans. Rob. Autom. 14(1)(1998) 166-171.