Overtaking Maneuvers by Non Linear Time Scaling over Reduced Set of Learned Motion Primitives

Vishakh Duggal[1], Kumar Bipin[1], Arun Kumar Singh[1], Bharath Gopalakrishnan[1], Brijendra Kumar Bharti[2], Abdelaziz Khiat[3] and K. Madhava Krishna[1]

***Abstract*— Overtaking of a vehicle moving on structured roads is one of the most frequent driving behavior. In this work, we have described a Real Time Control System based framework for overtaking maneuver of autonomous vehicles. Proposed framework incorporates Intelligent Planning and Modular control modules. Intelligent Planning module of the framework enables the vehicle to intelligently select the most appropriate behavioral characteristics given the perceived operating environment. Subsequently, Modular control module reduces the search space of overtaking trajectories through an SVM based learning approach. These trajectories are then examined for possible future time collision using Velocity Obstacle. It employs non linear time scaling that provides for continuous trajectories in the space of linear and angular velocities to achieve continuous curvature overtaking maneuvers respecting velocity and acceleration bounds. Further time scaling also can scale velocities to avoid collisions and can compute a time optimal trajectory for the learned behavior. The preliminary results show the appropriateness of our proposed framework in virtual urban environment.**

I. INTRODUCTION

The elementary goal in automating the driving process is to reduce accidents caused by human error and improve safety. The overtaking maneuver has been considered to be one of the toughest challenges in the development of autonomous vehicles. Autonomous vehicle requires three components for overtaking maneuver: high-level software framework to handle the complex task, generating overtaking trajectory based on perceived operating environment and producing control for vehicle satisfying physical geometry and kinematics constraints.

In this paper, we have described a Real Time Control System (RCS) [6] based framework for overtaking behavior of autonomous vehicles. It comprises of Intelligent Planning module and Modular control module. Intelligent Planning module was designed to address the issues associated with behavior mode selection in complex or unstructured environments of urban road. It enables the vehicle to intelligently select the most appropriate behavioral characteristics given the perceived operating environment. Intelligent Planning module composes of three main components Situation Assessment Specialist, Behavior Specialist and Decision Broker components, details in Section II-B.

[1]The authors are with Robotics Research Lab, IIIT Hyderabad-500032,Telengana,India. { *vishakh.duggal, kumar.bipin*}@research.iiit.ac.in, aks1812@gmail.com,bharath.gopalakrishnan@research.iiit.ac.in, mkrishna@iiit.ac.in.

[2]Brijendra Kumar Bharti is with Renault Nissan Technology and Business Center India Pvt Ltd, Chennai , India. *brijendrakumar.bharti@rntbci.com.*

[3]Abdelaziz Khiat is with Nissan Motor Co., Ltd, Japan. *khiat@mail.nissan.co.jp*.

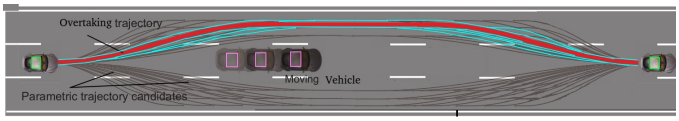

Fig. 1: Multiple Candidate overtaking trajectories: Trajectories (black) represent usual heuristic perturbation. Trajectories (blue) represents predicted collision free trajectories using machine learning and trajectory (red) means the selected optimum collision free.

The Modular control module uses Bezier polynomial [1] to model the motion of an autonomous vehicle, slow moving forward vehicle and other traffic vehicles with added Gaussian uncertainty [14]. A range of motion primitives for trajectories with high probability of being collision free are predicted using Machine Learning methodology and thereafter multiple candidate trajectories are perturbed using predicted motion primitives. These are then evaluated using Velocity Obstacle (VO) [2] to determine velocity scaling required for each selected trajectory to follow. The trajectory which provides least change in velocity profile is selected for generating non-linear motion profile with acceleration continuity while satisfying vehicle's dynamic constraints, details in Section III. Finally, we have demonstrated the applicability of the proposed method in Section IV, using UC-win/Road VR, a commercial package capable of simulating urban road environment and provides standard data structures for perception supported by Velodyne equipped automated vehicle [11].

In literature, modeling of autonomous vehicles overtaking behaviour in traffic simulation is presented in [10]. Wen Yao et al. [8] proposed predicting and selecting lane change trajectory based on real human driving data. Similarly, the future state of overtaking maneuver is predicted in research work by A Ghaffari et al. [9]. Petrov et al. provides insight into adaptive non-linear control of a vehicle for overtaking maneuver [4]. Generating optimum overtaking trajectory satisfying velocity, acceleration bounds constraints is presented [3].

The current paper presents the following novel contributions. Firstly it maps a perceived environment to most effective overtaking primitives through a SVM classifier thereby reducing the search space of possible overtaking primitives. Secondly on such learned primitives it computes a time optimal trajectory which satisfies collision avoidance, velocity bounds and continuity (including angular velocity) constraints through the time scaling method [15]. Thirdly it effectively integrates various modules through RCS architecture that seamlessly integrates higher level system software

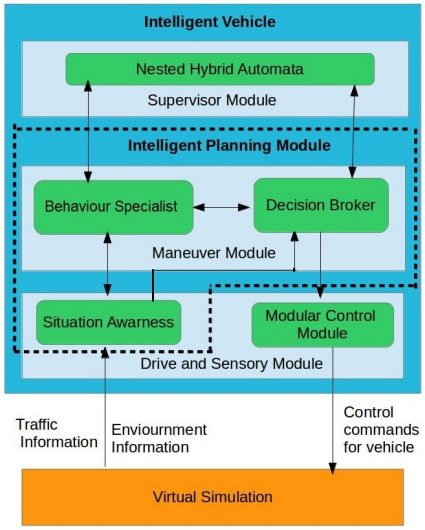

Fig. 2: Software architecture of the proposed framework implemented using RCS. RCS modules (light blue) shown with their respective autonomous overtaking software stack.

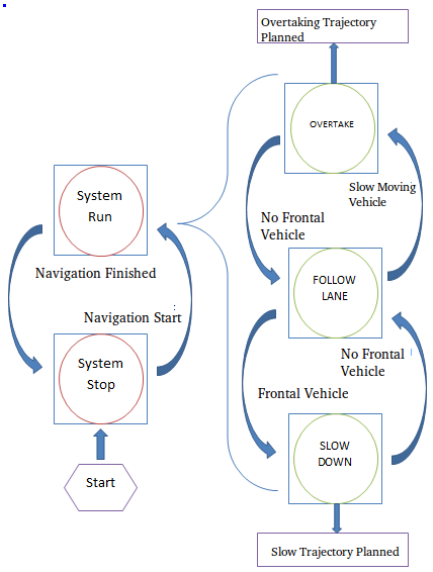

Fig. 3: Nested Hybrid Automaton of proposed framework.

with low level command and control modules. Specifically it differs from [3] in its ability to avoid vehicles on the side while overtaking the vehicle ahead in a time optimal fashion.

II. System and Software Architecture

The System and Software Architecture is implemented using Real Time Control System (RCS) which has communication, memory management and real time control infrastructure support. Real time response of the system is imperative for any application designed for autonomous vehicles as it ensures consistent and deterministic behavior. Fig. 2 shows the process view of system software architecture incorporated with RCS framework where each module represents separate process or task. Implementation is done using *rcslib* [6] library which is an open source and scalable implementation of RCS framework. The main modules of RCS framework are Supervisor module, Maneuver module and Drive and Sensory module.

A. Supervisor Module

To model the dynamic complexities of the on-road environments in the urban area, we choose to represent various situations a vehicle can be placed in using a Nested Hybrid automaton [7], where each state corresponds to a particular scenario, defined at different levels of abstraction. This approach leads to an elegant decomposition of very large problem into manageable sub-problems into a hierarchical structure. However, in a Nested Hybrid automaton, each state in the finite automaton represents either a discrete state (as in the hybrid automaton) or another sub-automaton which maintain a more fine-grained representation of the discrete state in a hierarchical manner. The advantages over a Standard Hybrid automaton [7] is an increased compactness of representation, modularity and isolation, as well as the ability to express discrete state transition at different levels of abstraction asynchronously. Our proposed design highest-level automaton consists of following two states :

- **System-Run** : Vehicle in motion and navigates autonomous through the urban environment.
- **System-Stop** : Stop the vehicle if in motion.

Fig. 3 illustrates this portion of the nested finite state automaton used to describe the process of urban driving. Where automaton defined within the System-Run state is nested and steps through the process of driving scenarios based on perceptual signals including the states: `[FOLLOW-LANE, SLOW-DOWN, OVERTAKE]`.

B. Maneuver Module

Maneuver Module of RCS framework consists of the Intelligent Planing module which addresses the issues associated with behavior mode selection in complex and unstructured environments of urban road. It enables the vehicle to intelligently select the most appropriate behavioral characteristics given the perceived operating environment. The framework is scalable to systems of varying complexity and size. It is composed of three principle elements tasked with assessing the situation, determining the suitability and viability of all possible solution and executing the most suitable of all recommended solutions. These three components are Situation Assessment, Behavior Specialist and Decision Broker.

1) Situation Assessment Component: Dynamic environment information, originating from any array of sensors is monitored and managed by the Situation Assessment Component. The inputs to the specialist can come from any data source. Whereas, findings are in the form of conditions, state or events. Once the findings have been generated the information is disseminated to all other components that might need it. Situation Assessment provided information used during the autonomous navigation as:`[CLEAR-VIEW, FRONT-VEHICLE-DETECTED SIDE-VEHICLE-DETECTED]`.

2) Behaviour Specialist Component: The findings rendered by the Situation Assessment Specialist are consumed

by the Behavior Specialist. The role of the specialist is to monitor the findings and evaluate the suitability of its behavior under the current perceived operating conditions. The specialist simultaneously monitors the desired travel lane, as well as adjacent travel lanes, for obstructions. Based on all inputs eg: vehicle speed, obstacle distance,max speed limit etc, the specialist recommends overtaking, slow down or continue motion as an appropriate options `[OVERTAKE, SLOW-DOWN, FOLLOW-LANE]`.

3) Decision-Broker Component: The Decision-Broker acts as a high level controller and monitor Behavior Specialist recommendation. It ensures recommended behaviour is executed by the vehicle in autonomous mode based on information and state obtained from Situation Assessment component as shown in Fig. 2. With each control iteration, if any vehicle is perceived in operating environment which is within the path of autonomous vehicle or within the desired safety following distance: Behavior Specialist component provides recommendation `SLOW-DOWN`, `OVERTAKE` or `FOLLOW-LANE` to the Decision-Broker component. Consequently, Decision-Broker component directs Modular Control module to generate optimum trajectory resulting overtaking, slowing down or follow lane maneuver for the autonomous vehicle.

C. Drive and Sensory Module

The Drive and Sensory Module mostly depends upon underlying perception and control interfaces of vehicle simulation tool. Such as Matlab simulation used for initial validation or TORCS, an open-source racing car simulator has limited interface. Whereas, UC-win/Road VR, a commercial package capable of simulating urban road environment and provides standard data structures for perception and control supported by Velodyne equipped automated vehicle. The perception provides information about states of static obstacles as well as moving traffic vehicles. Similarly, for control acceleration, velocity, steering angle and many more interfaces are available.

III. Design of an Optimal Overtaking Trajectory

In this section, our focus is specifically on the Modular Control module and algorithm for the overtaking maneuver. Following the direction from Decision-Broker component Modular Control module first, predicts the class of motion primitives for trajectories with high probability of being collision free using Machine learning methodology and thereafter multiple candidate trajectories are generated using predicted motion primitives as shown in Fig. 1. Secondly, Velocity Obstacle (VO) is exercised over all candidate trajectory to obtain optimum collision free trajectory and its respective velocity scaling factor, shown in Fig. 6.

A. Trajectory Set Generation

In simulated operating environment (velocity, acceleration, orientation, position) of traffic vehicles and autonomous vehicle are available. We use the Velocity Obstacle (VO) method to determine collisions with traffic vehicle in front. If collision is detected and if there is enough space to maneuver around the vehicle, we generate a set of N candidate trajectories each of the form given in (1), a quintic polynomial. In fact a family of such trajectories, $N = P \times Q$ in number are appropriately perturbed by uniform sampling intermediate states along both lateral P and longitudinal Q directions as mentioned in [8] and shown in Fig. 4.

$$P_i(\tau) = \sum_{i=0}^{n} \binom{n}{i} \tau^i (1-\tau)^{n-i} p_i, \forall \tau = 0, \ldots, 1 \tag{1}$$

where $\binom{n}{i}$ are binomial coefficient and $\binom{n}{i}\tau^i(1-\tau)^{n-i}$ are the *Bezier/Bernstein* basis polynomials, and p_i are scalar coefficient called *control points*. Solving trajectories primitives using (initial, intermediate, final) states and non-holonomic motion constraints, we obtain N candidate overtaking trajectories. In addition, bounded uncertainty in traffic vehicles motion is taken into consideration while predicting future time collision with autonomous vehicle [14]. This is a heuristic perturbation technique and computationally expensive for large value of N, among which many trajectories are not feasible. Therefore, optimizing heuristic perturbation of trajectories using Machine Learning approach is described in following section.

B. Machine Learning Data Set

Data set for learning optimized perturbation of trajectories for dynamic street scenario was generated in simulated environment using UC-win/Road [11] and TORCS [12]. N trajectories generated as mentioned above are partitioned into c classes with adjacent m trajectories belonging to same class. Multiple $(M = 680)$ complex scenarios are simulated and corresponding class of selected optimum collision free trajectory is recorded using approach described in [5]. Feature vector derived from the simulated environment for n traffic vehicles: $\{(x_1,y_1),\ldots,(x_n,y_n),d_1,d_2,\ldots,d_n,V_0,V_1,\ldots,V_n\}$ where V_0 is velocity of autonomous vehicle, (x_i,y_i), d_i, $V_i, \forall i = 1,\ldots,n$ being the relative position of i_{th} traffic vehicle with respect to autonomous vehicle, Euclidean Distance from autonomous vehicle and constant velocity of i_{th} traffic vehicle respectively. Feature vector are trained against their respective recorded class c_i of collision free trajectories using multi-class classification methodology. To achieve real-time performance *libSVM* Support Vector Machine package is used with training data set of size $680 * (n * 4 + 1)$. *SVM − RBF* provided real time performance with 72.5% accurate predictions. This contemporary design maps a perceived environment such as the one shown in Fig. 1 to a set of most likely motion primitives capable of efficient overtaking. In other words in the testing phase the feature vectors gleaned from the input perceived environment are mapped to a class c_i of trajectories that perform overtaking meanuevers.

C. Multiple Candidate Trajectories

Through the above method we are now able to accord a trajectory class c_i for a perceived environment characterized as feature vectors. Since c_i has $m \ll N$ trajectories, the search space for the best possible overtaking maneuver stands substantially reduced. It is to be noted that one is

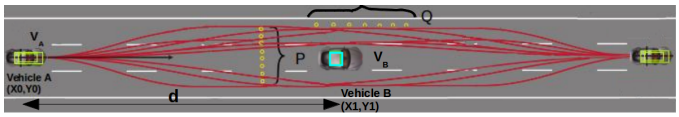

Fig. 4: Perturbation: Vehicle A (Autonomous vehicle) with position(X0,Y0), velocity V_A in collision with traffic vehicle Vehicle B with position(X1,Y1), velocity V_B and d relative distance between vehicles. Motion of vehicle is towards lateral direction and longitudinal direction perpendicular to it.

not guaranteed collision avoidance over these m trajectories. Hence collisions are checked using the method of Velocity Obstacle (VO). If collisions are detected in any of the m candidate trajectories, modification in the velocity profile is accomplished by scaling velocities that avoid collisions through the method described in [2]. Further the best possible scaling profile is computed that traverses the trajectory in optimal time for each of the candidate trajectory. The trajectory with least change in current state is further chosen as the trajectory to be executed. In both the cases collision with traffic vehicles is avoided. Our proposed approach provides computational efficiency in comparison to heuristic perturbation as presented in Table I.

D. Collision Free Trajectories

Optimum collision free trajectory needs to be selected from m candidate trajectories. We propose a *Velocity Obstacle* (VO) based solution for determining collision free trajectories. Let's consider the case of collision avoidance between autonomous vehicle in a collision course with moving forward traffic vehicle. $R = R1 + R2$, where $R1$ and $R2$ are safe distance radius of autonomous vehicle and traffic vehicle respectively. Collision is detected through the concept of Velocity Obstacle, which states that the autonomous vehicle and traffic vehicles are heading for a collision if the relative velocity vector $\vec{V}_{1/2}$ lies inside the collision cone $C_{1/2}$ of autonomous vehicle with respect to traffic vehicle depicted in Fig. 5. Collision is avoided if the relative velocity vector $\vec{V}_{1/2}$ is out of the collision cone $C_{1/2}$ which is given by the following condition [2]:

$$d^2 = |r^2| - \frac{\vec{r}.\vec{V}_{1/2}}{|\vec{V}_{1/2}|^2} \geq R^2 \quad (2)$$

$$\vec{r} = (x_1 - x_2)\hat{i} + (y_1 - y_2)\hat{j} \quad (3)$$

$$\vec{V}_{1/2} = (s\dot{x}_1 - \dot{x}_2)\hat{i} + (s\dot{y}_1 - \dot{y}_2)\hat{j} \quad (4)$$

The collision avoidance requires solving for the scale factor s from the following quadratic inequality.

$$(x_1 - x_2)^2 + (y_1 - y_2)^2 - R^2 - \frac{((s\dot{x}_1 - \dot{x}_2)(x_1 - x_2) + (s\dot{y}_1 - \dot{y}_2)(y_1 - y_2))^2}{(s\dot{x}_1 - \dot{x}_2)^2 + (s\dot{y}_1 - \dot{y}_2)^2} \geq 0 \quad (5)$$

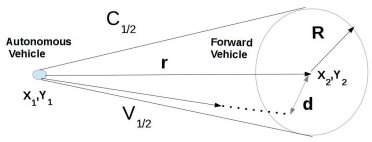

Fig. 5: Velocity Obstacle or Collision Cone

This inequality is solved for all m candidate trajectories for the traffic vehicle. Extending the collision avoidance from 1 traffic vehicle to n traffic vehicles involves finding n collision cones for each of m trajectories thus total $m*n$ computation are required. For each candidate trajectory, n constraints (5) which are in the form $a_i s^2 + b_i s + c_i > 0 \ \forall i = 1,2,\ldots n$ with roots $\alpha_i = \{-b_i - \frac{\sqrt{b_i^2 - 4a_i c_i}}{2a_i}\}$ and $\beta_i = \{-b_i + \frac{\sqrt{b_i^2 - 4a_i c_i}}{2a_i}\}$, are solved to obtain solution space of $s_{(i,j)}, \forall i = 1,\ldots,n \ and \ j = 1,\ldots,m$ [2]. For all candidate trajectories, intersection of respective n scaling factors $s_{(i,j)}$, $\forall i = 1,\ldots,n$ are calculated to obtain resultant scaling factor $s_{res(j)}$ which is outside collision cone of all n vehicles for the corresponding j^{th} trajectory. $s_{res(j)} \forall j = 1,\ldots,m$ represents corresponding resultant scaling factor of m candidate trajectories from which the optimum trajectory is selected. It can be noted that for $s_{res(j)} > 1$ robot needs to accelerate while for $s_{res(j)} < 1$ deacceleration is required. Hence trajectory with scaling factor $s_{res(j)}$ closest to 1 is selected as the next motion profile for autonomous vehicle.

TABLE I: Machine Learning efficiency against heuristic perturbation

No of Heuristic Trajectory Perturbation	Time (Sec)	Trajectory prediction using Machine Learning (Sec)	Computation saved %
21	0.1198	.0582	51.41
41	0.131297		55.67
61	0.174513		66.65
81	0.206515		71.81

The scaling factor s is obtained by the above method which is requisite for generating scaled velocity and acceleration profile based trajectory for the autonomous vehicle to perform overtaking maneuver, which will be discussed in subsequent section.

E. Time Scaling and Control Generation

We propose modifying the velocity and acceleration profile using time scaling based approach [2]. A change in the independent time variable from u to t in the trajectory definition $X(u)$ does not change the path taken by the robot, but brings the changes in the velocity and acceleration profile of the trajectory. This is the basis of time scaling of trajectory of the vehicle [5].

$$\dot{\mathbf{X}}(t) = \dot{\mathbf{X}}(u)\frac{du}{dt}, \ddot{\mathbf{X}}(t) = \ddot{\mathbf{X}}(u)(\frac{du}{dt})^2 + \dot{\mathbf{X}}(u)\frac{d^2u}{dt^2} \quad (6)$$

$\dot{X}(t)$, $\ddot{X}(t)$ represents individual velocity and acceleration components respectively. As it can be seen that the transformation just depends on the initial trajectory information. du/dt is the scaling function, which scales up the velocity and acceleration for $du/dt > 1$ and scales down for $du/dt < 1$. Since time cannot reverse itself, du/dt has to be a monotonic function. The total velocity v(t) and acceleration a(t) of a vehicle operating in 2D space can be written as:

$$v(t) = \sqrt{\dot{x}(t)^2 + \dot{y}(t)^2}, a(t) = \sqrt{\ddot{x}(t)^2 + \ddot{y}(t)^2} \quad (7)$$

$\dot{x}(t)$, $\ddot{x}(t)$ and similarly, others represent individual velocity and acceleration components respectively. Single exponential [2] and multiple exponential [5] based time scaling functions

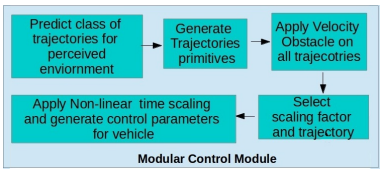

Fig. 6: Overtaking Maneuver: Flow chart.

are validated and found not dovetailed for overtaking trajectories. We propose the following parametric exponential based time scaling function [15] belonging to C^{∞} class of monotonic functions:

$$\dot{u} = e^{Q(u)} \Longrightarrow \ddot{u} = \dot{u}^2 \dot{Q}(u) \tag{8}$$

Where $Q(u)$ is l^{th} order Bezier polynomial in terms of variable u. In our formulation we have used $l = 31$, depending on the degree of smoothness required any suitable value can be selected. The transformation between the path and time variable needs to be a monotonic function. Using the notation $\dot{u} = du/dt$ and substituting (6) in (7) we get

$$v(t) = \dot{u}\sqrt{v(u)}, \quad and \quad v(u) = \dot{x}(u)^2 + \dot{y}(u)^2 \tag{9}$$

$$\dot{x} = v(t)\cos\theta, \quad and \quad \dot{y} = v(t)\sin\theta \tag{10}$$

where θ is heading angle and (10) represents the non-holonomic constraint of vehicle. Time optimal control along a specified path is achieved by bringing the velocity profile as close as possible to the velocity $v_{target} = s * v_{current}$ where $v_{current}$ is current velocity of the vehicle. In other words, the following objective function is to be minimized.

$$J = \int_{u_0}^{u_f} (\dot{u}\sqrt{v(u)} - v_{target})^2 \tag{11}$$

The objective function,(11) has to be minimized with respect to the following constraints

$$\dot{u}\sqrt{v(u)} \leq v_{target} \quad and \quad \dot{u} \in C^{\infty} \tag{12}$$

$$\dot{u}(u_0) = \dot{X}_0, \dot{u}(u_f) = \dot{X}_f, \quad and \quad \ddot{u}(u_0) = \ddot{X}_0, \ddot{u}(u_f) = \ddot{X}_f \tag{13}$$

$$|\ddot{x}(u)\dot{u}^2 + \dot{x}\ddot{u}| \leq \ddot{x}_{max}, \quad and \quad |\ddot{y}(u)\dot{u}^2 + \dot{y}\ddot{u}| \leq \ddot{y}_{max} \tag{14}$$

$$|\ddot{\theta}(u)\dot{u}^2 + \dot{\theta}\ddot{u}| \leq \ddot{\theta}_{max} \tag{15}$$

where $\ddot{x}_{max}$ and $\ddot{y}_{max}$ represent maximum acceleration constraints for vehicle in X and Y direction, and $\ddot{\theta}_{max}$ is maximum angular acceleration. The constraint (14) and (15) ensures that the linear and angular acceleration bounds are met. These are complex inequalities hence are difficult to solve due to their non-linear concave components. Hence a relatively simpler approach is adopted to obtain an approximate solution using discretization of time t. The time period $t \in (t_a, t_b)$ is divided in k intervals $t = (t_a, t_a + \delta, t_a + 2\delta, \ldots, t_b)$ and inequalities are solved at each of k^{th} interval boundary. Loss in optimality is attributed to discretization issues and can be reduced by increasing the resolution of discretization, although at the cost of increased computation time. Sequential convex programming **SCP** [13] routine is an efficient choice for solving above optimization problem due to its purely concave non-linear components [13]. In **SCP** procedure, at each iteration, the convex part of the problem is preserved while the non-convex part is replaced with a convex approximation around the solution obtained in the previous iteration. The resulting convex problem can be solved very efficiently and quickly. The **SCP** procedure is continued till some stopping criteria is met (Determined experimentally). (Details abstracted due to space constraints). Above formulation generates scaled velocity and acceleration profile of trajectory from which control inputs specific to the vehicle under consideration could be derived. These control inputs are applied in discrete time intervals for each control update.

IV. Experiment and Results

Conventional desktop computer equipped with an Intel Core i3 processor @ 2.2 GHz and 4GB of RAM was used for experiments. The simulation experiments were performed at initial validation stage on Matlab and later on TORCS [12] and UCWin [11] simulation software. Multiple runs for autonomous vehicle were carried out which was running with velocity range of $12 - 15m/s$ among several traffic vehicles moving with constant velocity between $5 - 8m/s$. Fig. 7(a) shows one of the overtaking behaviour of autonomous vehicle (blue) with forward traffic vehicle (green) while avoiding collision with side traffic vehicle (red) with uncertainty in position of traffic vehicle represented by larger bar representation as compared to autonomous vehicle. Time (T0,T1,T2,T3,T4) marked in blue, green and red represent autonomous vehicle(blue) and traffic vehicles(green and red) positions during overtaking maneuver at different times. Fig. 7(b) and Fig. 7(c) show continuity in velocity, acceleration and angular velocity respectively. Essentially this verifies the algorithm's capability to generate continuous curvature overtaking maneuvers that are also able to avoid other obstacles on the side while overtaking the vehicle in front on straight road. It is clearly seen that autonomous vehicle is able to overtake while avoiding collision. Similarly, Fig. 8(a) show slow down of the vehicle to avoid collision where overtaking may not be feasible. Fig. 9(b) depicts possible collision if velocity scaling is not performed with respective scaled and unscaled velocity profiles shown in Fig. 10. VideoLink: `http://youtu.be/OKHM6vqYlBU`

V. Conclusion and Future Works

This paper achieves a non-linear time scaling based continuous curvature overtaking maneuvers also capable of avoiding vehicles on the parallel lane. Further the best overtaking maneuver is selected from a reduced set of motion primitives of a trajectory class, this reduction from a substantially larger set achieved through a Machine Learning framework that maps an input environment characterized by feature vectors to a trajectory class. Efficient integration of the proposed algorithm with RCS framework and its further integration with state of the art simulators such as UC-Win and TORCS is another feature of this work. In future works, effects of road curvature and slope would be analysed.

VI. Acknowledgement

Financial supports received from M/S Renault Nissan Technology and Business Centre India Pvt. Ltd. (through a project entitled Driver Behavior Generation Using RCS Framework) and fruitful technical discussions with Hiroyuki Furushou, Nissan Motors Limited, Japan are thankfully acknowledged.

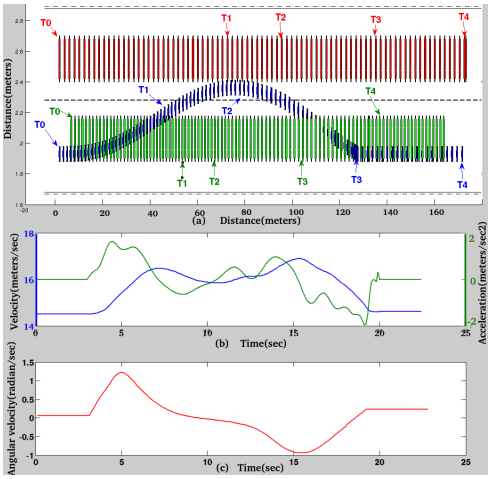

Fig. 7: Overtaking Scenario: (a) Motion profile of the autonomous vehicle(blue) with respect to traffic vehicles (red and green).Time (T0,T1,T2,T3,T4) in red, green blue show respective vehicular positions at respective times. (b) Velocity(blue) and acceleration(green) profile for autonomous vehicle. (c) Angular velocity of autonomous vehicle

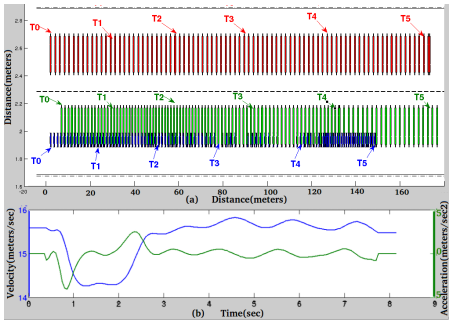

Fig. 8: Slowdown Scenario:(a) Motion profile of the autonomous vehicle(blue) with respect to traffic vehicles (red and green).Time (T0,T1,T2,T3,T4) in red, green blue show respective vehicular positions at respective times. (b) Velocity(blue) and acceleration(green) profile for autonomous vehicle.

References

[1] KG Jolly, R Sreerama Kumar, and R Vijayakumar. A bezier curve based path planning in a multi-agent robot soccer system without violating the acceleration limits. *Robotics and Autonomous Systems*, 57(1):23–33, 2009.

[2] Arun Kumar Singh and K Madhava Krishna. Reactive collision avoidance for multiple robots by non linear time scaling. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pages 952–958. IEEE, 2013.

[3] Tzila Shamir. How should an autonomous vehicle overtake a slower moving vehicle: Design and analysis of an optimal trajectory. *Automatic Control, IEEE Transactions on*, 49(4):607–610, 2004.

[4] Plamen Petrov and Fawzi Nashashibi. Modeling and nonlinear adaptive control for autonomous vehicle overtaking.

[5] Bharath Gopalakrishnan, Arun Kumar Singh, and K Madhava Krishna. Time scaled collision cone based trajectory optimization approach for reactive planning in dynamic environments. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 4169–4176. IEEE, 2014.

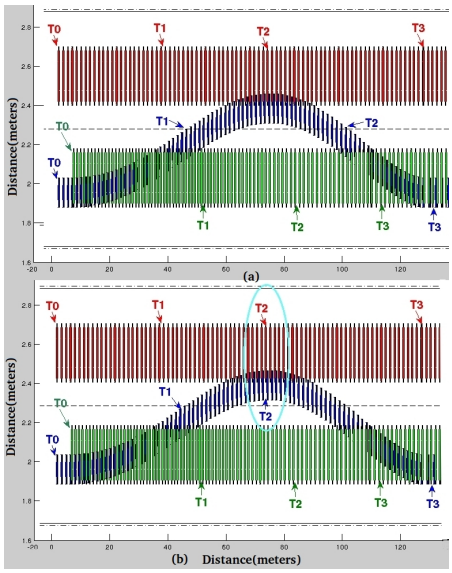

Fig. 9: Velocity Scaling (a) Collision free motion profile with velocity scaling (b) Possible collision prone motion profile without velocity scaling : Autonomous vehicle (Dark blue) with respect to traffic vehicles (red and green) and collision (eclipse light blue).Time (T0,T1,T2,T3) in red, green blue show respective vehicular positions at respective times.

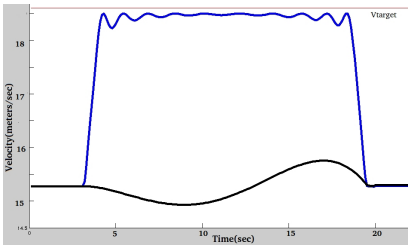

Fig. 10: Overtaking Scenario: (Black) unscaled velocity and (Blue) scaled velocity

[6] Kevin M Passino and Mathew L Moore. *The RCS Handbook: Tools for Real Time Control Systems Software Development*. John Wiley & Sons, Inc., 2001.

[7] Dave Wooden, Matt Powers, Magnus Egerstedt, Henrik Christensen, and Tucker Balch. A modular, hybrid system architecture for autonomous, urban driving. *Journal of Aerospace Computing, Information, and Communication*, 4(12):1047–1058, 2007.

[8] Wen Yao, Huijing Zhao, Franck Davoine, and Hongbin Zha. Learning lane change trajectories from on-road driving data. In *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pages 885–890. IEEE, 2012.

[9] A Ghaffari, A Khodayari, F Alimardani, and H Sadati. Manfis-based overtaking maneuver modeling and prediction of a driver-vehicle-unit in real traffic flow. In *Vehicular Electronics and Safety (ICVES), 2012 IEEE International Conference on*, pages 387–392. IEEE, 2012.

[10] Yue Yu, Abdelkader El Kamel, and Guanghong Gong. Modeling overtaking behavior in virtual reality traffic simulation system. In *Control Conference (ASCC), 2013 9th Asian*, pages 1–6. IEEE, 2013.

[11] Uc-win road vr simulation. `http://www.forum8.co.jp/english/uc-win/ucwin-road-e1.htm`. [Online 2015].

[12] Torcs-the open racing car simulator. `http://torcs.sourceforge.net/index.php`. [Online 2015].

[13] Stephen Boyd. Sequential convex programming. *Lecture Notes, Stanford University*, 2008.

[14] Maxim Likhachev and Dave Ferguson. Planning long dynamically feasible maneuvers for autonomous vehicles. *The International Journal of Robotics Research*, 28(8):933–945, 2009.

[15] Arun Kumar Singh. *Planning with Differential Constraints: Application to Navigation of Wheeled Mobile Robots on Uneven Terrain and in Dynamic Environments*. PhD thesis, International Institute of Information Technology, Hyderabad, India, September 2014. Chapter 5, Section 5.7.