Towards Intelligent Vehicles: Automatic Merge Control

Gurulingesh Raravi, Jatin Bharadia and Krithi Ramamritham
Indian Institute of Technology Bombay
Embedded Real-Time Systems Laboratory
{guru, jatin}@it.iitb.ac.in, {krithi}@cse.iitb.ac.in

Abstract

There is an increased concern towards the design and development of computer-controlled automotive applications to improve safety, reduce accidents, increase traffic flow, and enhance comfort for drivers. Automakers are trying to make vehicles more intelligent by embedding processors which can be used to implement Electronic and Control Software (ECS) for taking smart decisions on the road or assisting the driver in doing the same. These ECS applications are high-integrity, distributed and real-time in nature. Inter-Vehicle Communication and Road-Vehicle Communication (IVC/RVC) will only add to this intelligence by providing platform for distributed control implementation. Our work studies one such application, namely Automatic Merge Control System, which ensures safe vehicle maneuver in the region where two roads intersect. We have formulated this problem as an optimization problem aimed at minimizing the DTTI of vehicles, subject to certain constraints to ensure safety. In on-going work, we are implementing the system on robotic vehicular platforms built in our lab.

1. Introduction

It is believed that automation of vehicles will improve safety, reduce accidents, increase traffic flow, and enhance comfort for drivers. It is also believed that automation can relieve drivers from carrying out routine tasks during driving [1]. Automakers are trying to achieve automation by embedding more processors, known as Electronic Control Units (ECUs) and sensors which help enhance the intelligence of the vehicle. This processing power can be utilized effectively to make an automobile behave in a smart way, e.g., by sensing the surrounding environment and performing necessary computations on the captured data either to decide and give commands to execute the action or to assist a driver in taking decisions. In modern day automobiles, several critical vehicle functions such as vehicle dynamics, stability control and powertrain control, are handled by ECS applications.

Adaptive Cruise Control (ACC) is one such intelligent feature that automatically adjusts vehicle speed to *maintain the safe distance* from the vehicle moving ahead on the same lane (a.k.a. *leading vehicle*). When there is no vehicle ahead, it tries to maintain the safe speed set by the driver. Since ACC is a safety-enhancing feature it also has stringent requirements on the freshness of data items and completion time of the tasks. The design and development of centralized control for ACC with efficient real-time support is discussed in [2].

The design and development of ACC application should also focus on increasing fuel efficiency, traffic flow, etc. To help achieve these goals, modern vehicles are equipped with wireless communication for supporting IVC/RVC. The effort of making effective use of wireless communication is in progress by realizing inter-vehicle communication and vehicle road-side infrastructure communication to make the whole transport system intelligent. Sophisticated distributed control features having more intelligence and decision making capability like collision-avoidance, lane keeping and by-wire systems are on the verge of becoming a reality. In all such applications, wireless communication provides the flexibility of having distributed control. A distributed control system brings in more computational capability and information which helps in making automobiles more intelligent. In this paper, we focus on one such distributed control application, namely Automatic Merge Control System which tries to ensure safe vehicle maneuver in a region where two roads intersect. To this end, we have formulated an optimization problem with the objective to minimize the maximum *driving-time-to-intersection (DTTI)* (time taken by vehicles to reach the intersection region) subject to specific safety-related constraints.

The rest of the paper is organized as follows. Section 2 introduces *Automatic Merge Control System* and describes the problem in detail. The optimization function and constraints are formulated in Section 3. The results of Matlab-based evaluations are discussed in Section 4. Section 5 presents the conclusions, related work and future work.

2. Automatic Merge Control System

The Automatic Merge Control (AMC) System is a distributed intelligent control system that ensures safe vehicle maneuver at road intersections. The system ensures that no two vehicles coming from different roads collide or interfere at the intersection region. It ensures that the time taken by any two vehicles to reach the intersection region is separated by a timegap, at least δ (which depends on the length of the intersection region and velocity of vehicles), by giving commands to adapt their velocities appropriately. In other words, it ensures that no two vehicles will be present in the intersection region at any given point of time.

Our goal is to formulate this as an optimization problem to minimize the maximum *DTTI*. *DTTI* of a vehicle is the time it takes to reach the intersection region. We have made following assumptions while formulating the optimization problem.

- An intelligent (communication + computation) infrastructure node is situated road-side near the intersection region. It performs all computations and determines the commands (acceleration, deceleration) to be given to each vehicle.
- The communication infrastructure enabling the vehicles on the road and infrastructure node to communicate with each other exists.
- All vehicles are equipped with ACC feature to maintain safe distance from their respective leading vehicle.
- Each vehicle has an intelligent control application which takes velocity, distance and time as input and ensures that the vehicle will travel that distance in given time and the vehicle velocity when it finishes traveling the given distance will be equal to the given velocity.
- The vehicles inside the *Area of interest* (AoI) becomes part of the system i.e., their profiles(velocity, acceleration and distance) will be tracked by infrastructure node and it can also give commands to those vehicles to accelerate or decelerate.

3. Specification of the DTTI Optimization Problem

We first take up the simple case of two roads merging and then extend it to more than 2 roads.

3.1. Two-Road Intersection

In this section, we give the formulation of the optimization function subject to constraints ensuring their safety. Consider an intersection of two roads, $Road_1$ and $Road_2$

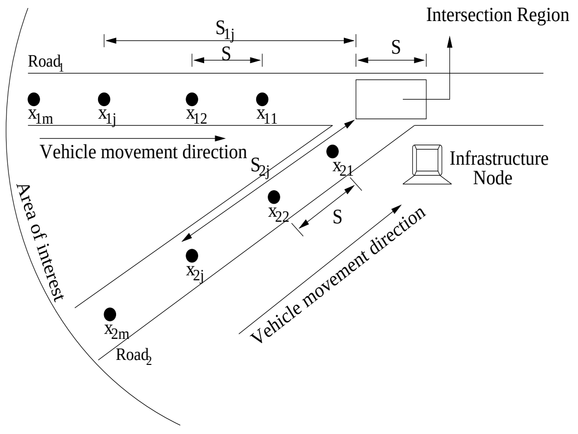

Figure 1. Automatic Merge Control System

as shown in Figure 1 where vehicles are represented by points. It is assumed that each road contains m_i vehicles.

For the rest of this section the range of i and j are given by, $1 \leq i \leq 2$ (represents road index) and $1 \leq j \leq m_i$ (represents vehicle index) unless otherwise specified explicitly. Table 3.1 describes the notations used in the formulation.

Notation	***Description***
$Road_i$	represents i^{th} road
m_i	number of vehicles in $Road_i$
x_{ij}	j^{th} vehicle on $Road_i$
s_{ij}	distance of the vehicle x_{ij} from the intersection region
u_{ij}	initial velocity of the vehicle x_{ij}
t_{ij}	time at which the vehicle x_{ij} reach the intersection region
DoS(x_{ij}, x_{ij+1})	Distance of Separation between j^{th} vehicle and $(j+1)^{th}$ vehicle on $Road_i$ where $1 \leq j \leq m_i - 1$

Table 1. Notations used in the formulation

- **Objective Function:** The objective is to minimize the maximum DTTI (i.e., time taken by the vehicle say x_{im_i} to reach the intersection region).
  $Minimize f = MAX(t_{1m_1}, t_{2m_2})$
  This is similar to the *makespan* of a schedule. An alternative is to minimize the average DTTI:
  $$Minimize f = \frac{1}{m_1+m_2} * (\sum_{i=1}^{m_1} t_{1m_i} + \sum_{i=1}^{m_2} t_{1m_i})$$
- **Precedence Constraint:** This constraint is to ensure that the vehicles within a road reach the intersection region according to the ascending order of their distance from the region i.e., no vehicle overtakes its leading vehicle. The ACC present in individual vehicles ensures that the DoS criteria are never violated between consecutive vehicles.
  For $Road_i$, $t_{ij} \leq t_{ij+1}$ where $1 \leq j \leq m_i - 1$.

- **Mutual Exclusion Constraint:** This guarantees that no two vehicles are present in the intersection region at any given point of time. In other words, this condition ensures that before $(j+1)^{th}$ vehicle reaches the intersection region, the j^{th} vehicle would have traveled through the region.
  For $Road_i$, $t_{ij+1} \geq t_{ij} + \frac{S}{v_{ij}}$, where $1 \leq j \leq m_i - 1$. The above condition guarantees that no vehicles from same road will be present in the intersection region. To ensure vehicles from different roads also adhere to this safety criterion we have:
  $\forall k, l(|t_{1k} - t_{2l}| \geq \frac{S}{min(v_{1k}, v_{2l})})$
  where k and l represents vehicle index numbers and v_{1k} and v_{2l} are the velocities of vehicles x_{1k} and x_{2l} when they reach the intersection region respectively.

- **Lower bound on Time:** This imposes lower bound on the time taken by any vehicle to reach intersection region with the help of V_{MAX}, maximum velocity any vehicle can attain.
  For $Road_i, \forall j \;\; t_{ij} \geq \frac{s_{ij}}{V_{MAX}}$.

- **Equality Constraint on Velocity:** This constraint relates the velocity of vehicle at the intersection region to its initial velocity, the distance traveled and the time taken to do so.
  For $Road_i, \forall j \;\; v_{ij} = \frac{2s_{ij}}{t_{ij}} - u_{ij}$.

After replacing all v_{ij} in the above set of constraints using the equality constraint on velocity, the system is left with the following design variable(s): t_{ij}.

System Input: $\forall i, j \;\; s_{ij}, u_{ij}$, S, and V_{MAX}.

System output: $\forall i, j \;\; t_{ij}$.

The acceleration or deceleration commands to be given to each vehicle can be computed offline from the output of system using:

$$\forall i, j \;\; a_{ij} = \frac{2 * (s_{ij} - u_{ij} * t_{ij})}{t_{ij}^2} \qquad (1)$$

3.2. n-Road Intersection

In this section, we provide the formulation for a case where *n* roads are intersecting. The formulations in Section 3.1 can be easily extended to suit this scenario.

For the rest of this section the range of i and j are given by, $1 \leq i \leq n$ (represents road index) and $1 \leq j \leq m_i$ (represents vehicle index) unless otherwise specified explicitly. Similarly, range for k and l are given by, $1 \leq k \leq n$ (represents road index) and $1 \leq l \leq m_k$ (represents vehicle index).

- **Objective Function:**
  (1). $Minimize f = \forall i \; MAX(t_{im_i})$ OR
  (2). $Minimize f = \frac{1}{\sum_{i=1}^{n} m_i} * (\sum_{i=1}^{n}\sum_{j=1}^{m_i} t_{ij})$

- **Precedence Constraint:**
  $\forall i \;\; t_{ij} \leq t_{ij+1}$ where $1 \leq j \leq m_i - 1$

- **Mutual Exclusion Constraint:**
  $\forall i, j, k, l \;\; |t_{ij} - t_{kl}| \geq \frac{S}{min(v_{ij}, v_{kl})}$

- **Lower bound on Time:** $\forall i, j \;\; t_{ij} \geq \frac{s_{ij}}{V_{MAX}}$

- **Equality Constraint on Velocity:**
  $\forall i, j \;\; v_{ij} = \frac{2s_{ij}}{t_{ij}} - u_{ij}$.

System Input: $\forall i, j \;\; s_{ij}, u_{ij}, S$, and V_{MAX}.

System output: $\forall i, j \;\; t_{ij}$.

As can be observed from the above formulation, there is not much difference between our 2-road and n-road formulations.

4. Matlab-based Evaluation Results and Observations

This section describes the Matlab-based evaluations carried out and observations that can be made from these. For simplicity, we considered a 2-road intersection problem where each road is having two vehicles. We modeled this system in MATLAB using its Optimization Toolbox (function *fmincon*). The inputs provided to the model are:

$$u = \begin{pmatrix} 3 & 5 \\ 5 & 6 \end{pmatrix}, \;\; s = \begin{pmatrix} 6 & 10 \\ 10 & 12 \end{pmatrix}, \; S = 2, \; V_{MAX} = 7$$

i.e., initial velocities of vehicles x_{11} and x_{12} are set to $3m/s$ and $5m/s$ and their distances from the intersection region are set to $6m$ and $10m$ respectively. Similarly, for vehicles x_{21} and x_{22} velocities are set to $5m/s$ and $6m/s$ and distances are set to $10m$ and $12m$. Length of intersection region is set to $2m$ and maximum velocity any vehicle can attain is set to $7m/s$.

The model came up with the time at which each vehicle is allowed to enter the intersection region as output which is depicted in Figure 2. The X-axis represents the time and Y-axis indicates the distance of each vehicle from region of intersection (i.e., $-x$: vehicle needs to travel x distance to reach the intersection region, 0: vehicle has reached the region and $+x$: distance covered by vehicle after leaving the region). It can be observed that our model guarantees that when x_{ij} reaches the region of interest the distance between it and vehicle in front of it, say x_{kl}, is at least S. It can also be observed that the curves are quadratic in nature falling in-line with the quadratic equation of motion (see Equation -1). The limitation of our model can be observed in the graph i.e., our model does not guarantee that safety

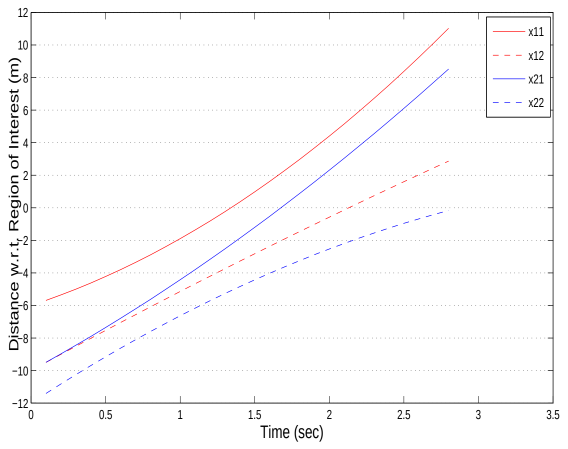

Figure 2. AMCS: 2-road 2-vehicle Scenario

distance criteria hold true at every instant of time (remember we had stated this as one of the assumptions in Section 2: each vehicle is equipped with ACC which takes care of this condition). It should also be observed that the model shown does not provide the safe distance guarantee after the region of interest. But we can incorporate other region of interests in the model by adding few more constraints in the same model.

4.1. Using the output of the Optimizer

In a real world, the above optimization problem can be solved sporadically for snapshots of the area of interest by making a realistic assumption that the minimum time that any vehicle takes to enter the AoI is known. The frequency of execution of this algorithm is driven by following parameters: x, the distance of the closest outside vehicle from the AoI boundary and V_{MAX}, maximum velocity any vehicle can attain. Hence, the closest vehicle will take at least x/V_{MAX} time to enter AoI. The sporadicity of this algorithmic task can be determined by imposing a lower bound on x.

5. Conclusions and Further Work

In this paper, we presented Automatic Merge Control System that ensures safe vehicle maneuver at road intersections. We formulated this as an optimization problem with constraints to guarantee safety. It is shown with the help of MATLAB Optimization Toolbox that the existing constraint solvers can be used to determine the solution. Our model guarantees safe distance criteria only at the intersection region assuming that individual vehicles are equipped with smart functionalities to maintain safe distance between them and reaching the destination in given time at given velocity.

The merge control application with inter-vehicle communication is also studied in [3]. It uses the concept of *virtual vehicle* that is used to map vehicles in one lane onto the other lane (assuming a 2-lane merge) for ensuring safe distance criteria. But the algorithm for determining the merge order of vehicles is not provided.
The intersection region is divided into multiple zones in [4] where suboptimal velocity profiles computed for vehicles initially gets refined to optimal profiles as the vehicles approach the zone nearer to intersection point. The approach requires more processing power in every vehicle compared to ours since each vehicle computes the merge order. The communication overhead is also more since every vehicle communicates with all the nearby vehicles about its profile. Also, we believe our formulation is simple to understand and implement.

Our future plan for this on-going work is as follows. First, extend the system to ensure safe distance criteria at every instant of time. By defining *Point of Interests* on roads at regular intervals and taking decision about accelerating and decelerating at every point will tackle this problem to a certain extent. Second, provide real-time support for the system and demonstrate the concept on robotic vehicular platforms built in our lab. Third, reformulating the objective function and constraints for different optimization functions such as maximizing throughput, minimizing the cost considering several factors like priority of vehicles, priority of roads, angle of intersection of roads, etc. Fourth, applying approximation algorithm techniques to solve the problem.

References

[1] A. Vahidi and A. Eskandarian, “Research advances in intelligent collision avoidance and adaptive cruise control,” *IEEE Transactions on Intelligent Transportation Systems*,, vol. 4, pp. 143–153, Sept. 2003.

[2] G. Raravi, N. Sharma, K. Ramamritham, and S. Malewar, “Efficient real-time support for automotive applications: A case study,” in *RTCSA ’06: Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, Sydney, Australia, 2006, pp. 335–341.

[3] T. Uno, A. Sakaguchi and S. Tsugawa, “A merging control algorithm based on inter-vehicle communication,” in *IEEE International Conference on Intelligent Transportation Systems*, Tokyo, Japan, 1999, pp. 783–787.

[4] T. Bruns and E. Munch, “Intersection management as self-organisation of mechatronic systems,” in *Proceedings of 6th International Heinz Nixdorf Symposium on New Trends in Parallel and Distributed Computing*, Paderborn, Germany, Jan. 2006.