MERGE ALGORITHMS FOR INTELLIGENT VEHICLES

Gurulingesh Raravi, Vipul Shingde, Krithi Ramamritham, Jatin Bharadia
Embedded Real-Time Systems Lab
Indian Institute of Technology Bombay
{guru, jatin}@it.iitb.ac.in, {vipul, 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) mechanisms will only add to this intelligence by enabling distributed implementation of these applications. 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 discussed two approaches for designing this system both aimed at minimizing the *Driving-Time-To-Intersection* (DTTI) of vehicles, subject to certain constraints for ensuring safety. We have (i) formulated this system as an optimization problem which can be solved using standard solvers and (ii) proposed an intuitive approach namely, Head of Lane (HoL) algorithm which incurs less computational overhead compared to optimization formulation. Simulations carried out using Matlab and C++ demonstrate that the proposed approaches ensure safe vehicle maneuvering at intersection regions. In this on-going work, we are implementing the system on robotic vehicular platforms built in our lab.

Keywords: Automatic merge control, Driving-time-to-intersection, Area-of-interest, Vehicle merge sequence, Vehicle interference, Continuous vehicle stream

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 [Vahidi and Eskandarian, 2003]. Automakers are trying to achieve automation by embedding more processors, known as Electronic Control Units (ECUs) and sensors into vehicles which help to enhance their intelligence. 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 carry out the necessary action or to assist the 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 [Raravi et al., 2006].

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 *n* roads intersect. To this end, we have (i)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 and (ii) proposed *Head of Lane (HoL)* algorithm for achieving the same with less computational overhead compared to optimization formulation.

In this paper, terms *road* and *lane* are used interchangeably. The rest of the paper is organized as follows. Section 1 introduces *Automatic Merge Control System* and describes the problem in detail. The optimization function and constraints are formulated in Section 2. The HoL algorithm is described in Section 3. The results of simulation and Matlab-based evaluations are discussed in Section 5. Section 6 presents the related work followed by conclusions and future work.

1. 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 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 instant of time. This system involves: (i) determining the *Merge Sequence (MS)* i.e., order in which vehicles cross the intersection region (ii) ensuring safety at intersection region and (iii) achieving an optimization goal such as minimizing the maximum *(DTTI*, time taken by a vehicle to reach the intersection region.

Our goal is to ensure safe vehicle maneuver at intersection regions which involves the above mentioned three subproblems.

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.
- A suitable communication infrastructure exists for vehicles and roadside infrastructure node to communicate with each other.
- Initially, all the vehicles are atleast *S* distance apart (safety distance) from their respective leading vehicle.
- Each vehicle has an intelligent control application which takes acceleration and time as input and ensures that the vehicle reaches the merge region in that time periods by following the given acceleration.
- Only those vehicles which are inside the *Area of interest* (AoI) are part of the system i.e., their profiles(velocity, acceleration and distance) will be tracked by roadside infrastructure node and commands can be given to those vehicles to accelerate or decelerate.

2. 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.

2.1 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 $Road_2$ as shown in Figure 1 where vehicles are represented by points. It is assumed that $Road_i$ contains m_i vehicles where $1 \leq i \leq 2$.

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

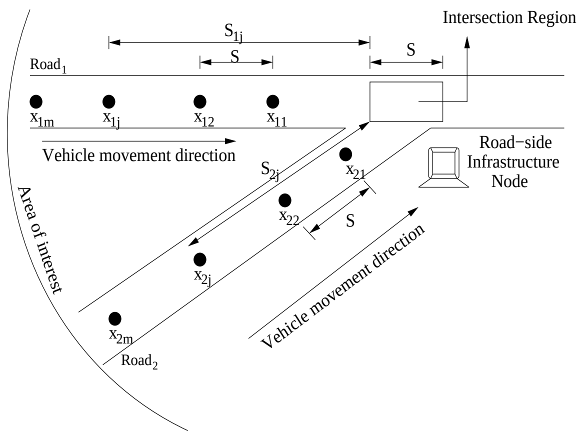

Figure 1. Automatic Merge Control System

otherwise specified explicitly. Table 1 describes the notations used in the formulation. These notations will be used throughout the paper.

Table 1. 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 $Road_i$
$s_{ij}(t)$	distance of the vehicle x_{ij} from the intersection region at time instant t
u_{ij}	initial velocity of the vehicle x_{ij}
v_{ij}	velocity of the vehicle x_{ij} when it reaches the merge region
t_{ij}	time at which the vehicle x_{ij} reaches the intersection region

- **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: $Minimize f = \frac{1}{m_1+m_2} * (\sum_{i=1}^{m_1} t_{1i} + \sum_{j=1}^{m_2} t_{2j})$

- **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:
  For $Road_i$, $t_{ij} < t_{i(j+1)}$ 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 instant 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 $Road_i$, $t_{i(j+1)} \geq t_{ij} + \frac{S}{v_{ij}}$, 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
  $\forall k, l(|t_{1k} - t_{2l}| \geq \frac{S}{v})$
  where v will take value v_{1k} or v_{2l} depending on whether $t_{1k} < t_{2l}$ or $t_{2l} < t_{1k}$ respectively and k and l represent vehicle index numbers.

- **Safety Constraint:** This constraint ensures that safe distance is always maintained between consecutive vehicles on the same road, before they enter the merge region. Consider two such consecutive vehicles x_{ij} and $x_{i(j+1)}$ on $Road_i$. For safety, the following condition needs to be ensured: $\forall t \in (0, t_{ij})$, $s_{i(j+1)}(t) - s_{ij}(t) > S$.

  Distance between x_{ij} and $x_{i(j+1)}$ is given by:
  $s_{i(j+1)}(t) - s_{ij}(t) = (s_{i(j+1)}(0) - (u_{i(j+1)} * t + *a_{i(j+1)} * t^2)) - (s_{ij}(0) - (u_{ij} * t + *a_{ij} * t^2)) = f(t)$

  Ensuring $f_{min}(t) > S$ will guarantee safety criteria. On simplification, the following constraint is obtained:

  ```
  For $Road_i$,$\forall j$
  **if** ($a_{ij} > a_{i(j+1)}$ and $(u_{ij} - u_{i(j+1)})/(a_{i(j+1)} - a_{ij}) < t_{ij}$) **then**
     $s_{i(j+1)}(0) - s_{ij}(0) - L > (u_{ij} - u_{i(j+1)})^2/(2 * (a_{ij} - a_{i(j+1)}))$
  **else**
     Mutual Exclusion Constraint guarantees that the safety criteria will be satisfied.
  ```

- **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 $Road_i, \forall j \;\; t_{ij} \geq \frac{s_{ij}}{V_{MAX}}$ where s_{ij} is the initial distance from intersection region i.e., at time instant $t = 0$. Throughout the paper, s_{ij} and $s_{ij}(0)$ are used interchangeably.

- **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 $Road_i, \forall j \;\; v_{ij} = \frac{2s_{ij}}{t_{ij}} - u_{ij}$.

- **Other Constraints:** These constraints impose limits on the velocity and acceleration range of vehicles.
  For $Road_i, \forall j \;\; V_{MIN} \leq v_{ij} \leq V_{MAX}; A_{MIN} \leq a_{ij} \leq A_{MAX}$

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:

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

2.2 n-Road Intersection

In this section, we provide the formulation for a case where *n* roads are intersecting. The formulations in Section 2.1 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:** $\forall i \;\; t_{ij} < t_{ij+1}$ where $1 \leq j \leq m_i - 1$
- **Mutual Exclusion Constraint:**
  $\forall i, j, k, l \;\; |t_{ij} - t_{kl}| \geq \frac{S}{v}$ where v will take value v_{ij} or v_{kl} depending on whether $t_{ij} < t_{kl}$ or $t_{kl} < t_{ij}$ respectively.
- **Safety Constraint:**
  $\forall i, j$ if ($a_{ij} > a_{i(j+1)}$ and $(u_{ij} - u_{i(j+1)})/(a_{i(j+1)} - a_{ij}) < t_{ij}$) then
  $s_{i(j+1)}(0) - s_{ij}(0) - L > (u_{ij} - u_{i(j+1)})^2/(2 * (a_{ij} - a_{i(j+1)}))$
- **Lower bound on Time:** $\forall i, j \;\; t_{ij} \geq \frac{s_{ij}}{V_{MAX}}$
- **Equality Constraint on Velocity:** $\forall i, j \;\; v_{ij} = \frac{2s_{ij}}{t_{ij}} - u_{ij}$.
- **Other Constraints:**
  $\forall i, j \;\; V_{MIN} \leq v_{ij} \leq V_{MAX}; A_{MIN} \leq a_{ij} \leq A_{MAX}$

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.

3. Head of Lane Approach

In this section, we describe another algorithm for determining the merge sequence. We discuss the case of two roads merging at an intersection while the work of extending it to n-roads merging is in progress. This approach is motivated by the way drivers in manually driven vehicles resolve the conflict at intersection region in practice. The drivers who are closest to the merge region on each road decide among themselves the order in which they will pass through the region (based on some criteria, say *First Come First Serve*).

This algorithm achieves the goal of safe maneuvering by considering the foremost vehicles on each lane for determining the merge sequence. This approach incurs lesser computational overhead compared to optimization formulation and easily maps to the way merging happens in real-world scenario where vehicles are not automated. The algorithm is explained in detail below for two roads merging scenario.

3.1 Two-Road Intersection

Consider the scenario depicted in Figure 1, where x_{11} and x_{21} are head vehicles (vehicles nearest to merge region) on $Road_1$ and $Road_2$ respectively whose DTTI are conflicting and hence are the competitors for the same place in MS. The algorithm resolves the conflict among these two vehicles by computing the cost associated with each vehicle (determining this cost is explained in Section 3.3) and adding the one with the lower cost, say x_{21} in the MS. Now, the algorithm considers the head vehicles on each road: x_{11} from $Road_1$ and x_{22} from $Road_2$ (since x_{21} is already included in the MS, x_{22} is the current head vehicle on $Road_2$), resolves conflict, adds the vehicle with minimum cost in MS and so on. This is done iteratively till all the vehicles are merged.

HoL algorithm operates with the same set of constraints formulated in Section 2. The goal of optimization formulation was to achieve minimum average DTTI or maximum throughput. HoL too tries to achieve the same goal by employing *acceleration whenever possible* approach. For example, in the scenario explained above, x_{21} is assigned maximum possible acceleration before inserting it in the merge sequence. A single iteration of the HoL algorithm is discussed in detail below:

1 Let x_{1k} and x_{2l} be the two head vehicles in a particular iteration. In the first iteration the foremost vehicles on each lane (x_{11} and x_{21}), would be the head vehicles.

2 Depending upon the behavior of vehicles in the MS, algorithm determines the future behavior B_{1k}, B_{2l} of the vehicles x_{1k}, x_{2l} respectively. Here future behavior of a vehicle represents its kinematics from current time instant till the vehicle reaches the merge region. While determining these behaviors, the vehicles are accelerated whenever possible while ensuring that all the constraints are met. Note that, B_{1k} and B_{2l} are calculated independent of one another and can conflict during/after merging.

3 Verify whether the behavior B_{1k} and B_{2l} interfere in the merge region (explained in detail in Section 3.2), i.e., whether the vehicles violate the safety criteria in the merge region.

4 If they are not interfering then insert that vehicle in the MS which is reaching the region M first, say x_{1k}. If they are interfering then compute the cost c_1 of the merge sequence (determining cost will be explained in Section 3.3) in which x_{1k} is chosen to be inserted in MS. Similarly compute cost c_2 in which x_{2l} is chosen to be inserted. Compare cost c_1 and c_2 and insert the vehicle with lower cost in the MS.

5 Depending upon which vehicle has been inserted in the MS, say x_{1k}, consider $x_{1(k+1)}$ and x_{2l} as head vehicles for the next iteration. Similarly if x_{2l} is inserted, then consider x_{1k} and $x_{2(l+1)}$ as head vehicles.

3.2 Interference in Merge Region

The head vehicles x_{1k} and x_{2l} from $Road_1$ and $Road_2$ still have the possibility of violating the safety criteria in the merge region even after determining their future behavior B_{1k} and B_{2l} respectively, as the behaviors are computed independent of one another. This violation of safety criteria in the merge region is called *vehicle interference*. The vehicles might be strongly violating the safety criteria, i.e. both the head vehicles might be entering the merge region approximately at same time. In this case, resolving the conflict becomes slightly tricky and the algorithm must choose the vehicle with lower cost. While in another scenario, the vehicles might be violating the safety criteria by a very small amount, i.e. when a vehicle is just about to exit the merge region, another vehicle might enter it. This special case is handled in a similar way as the non-interference one, where the leading head vehicle is inserted in the MS. We differentiate these two cases as described below:

- **Vehicle Interference** $(|t_{1k} - t_{2l}| < \delta)$: Determine cost c_1, of the merge sequence in which x_{1k} is chosen to be added first to MS. Similarly determine cost c_2 for adding x_{2l}. Insert that vehicle in the MSwhich has lower cost associated with it.
- **Non Interference** $(|t_{1k} - t_{2l}| > \delta)$: If $t_{1k} > t_{2l}$ then insert x_{2l} in MS else insert x_{1k} in MS.

The value of δ can be determined using safety distance (S) and velocity of the head vehicles.

3.3 Merge Cost Computation

When two vehicles strongly interfere (in the merge region) and compete for the same place in MS, the HoL approach described above computes the cost of inserting each vehicle in the MS at that particular place and resolves the conflict by choosing the one with lower cost. Here we describe two approaches for determining this cost (associated with a particular vehicle for inserting it in the MS). The first approach has been simulated and work is in progress on simulation of the second approach.

- **Nearest Head:** In case of strong interference, both the head vehicles take almost same time to reach the merge region. It is more reasonable to allow the vehicle which is closer to the merge region to go first as it will have lesser time to adapt to any changes (deceleration). If both the vehicles are equidistant from the merge region, then algorithm randomly chooses one of them.
- **Cascading Effect:** This approach considers the effect on previous vehicles on each road, while computing the cost for resolving the conflict. This effect can be measured in terms of net deceleration introduced, the number of vehicles that are being affected as both give a measure of increase in DTTI of vehicles. Optimal approach would be to consider all possible merge sequences and choose the best among them. Though this solution is better in terms of optimality, but it will be computation intensive, as in this case the total number of merge orders considered will be exponential.

3.4 Pseudo Code

Pseudo code of HoL algorithm is presented in detail below. All the notations conform to the notation used in section 2. Functions used in the pseudo-code are explained below:

1 **getBestPossibleBehavior(Profile P, Merge Sequence MS)**: It takes profile(P) of a vehicle and Merge Sequence(MS) as input and with the

help of behavior of vehicles that are in the MS, the function determines (and returns) the best possible future behavior for that vehicle.

2 **computeTimeToReach(Behavior B)**: It takes future behavior(B) of a vehicle as input and then computes (and returns) DTTI of that vehicle.

3 **checkStrongInterference(Behavior B_1 , Behavior B_2, safe distance S, InterferenceParameter δ)**: It takes behavior(B_1 and B_2) of two vehicles and system parameters: S and δ as input and then determines whether these vehicles interfere in the merge region. An appropriate boolean value is then returned (1 - if they interfere, 0 - otherwise).

```
HoL Algorithm Begin
  $k = l = 1$;
  while ($k <= m_1$ and $l <= m_2$){
     $B_1$ = getBestPossibleBehavior( $P_{1k}$, $MS$ );
     $B_2$ = getBestPossibleBehavior( $P_{2l}$, $MS$ );
    //Where $P_{ij}$ is the current profile(velocity, acceleration and distance
     //from region $M$) of vehicle $x_{ij}$.
     $t_1$ = computeTimeToReach( $B_1$ );
     $t_2$ = computeTimeToReach( $B_2$ );
     StrongInterference = checkStrongInterference( $B_1$ , $B_2$, $S$, $\delta$ );
     if ( StrongInterference ) then{
         Determine the cost $c_1$ and $c_2$;
         if($c_1 < c_2$) then
              Insert $x_{1k}$ in $MS$; $k = k + 1$;
         else
              Insert $x_{2l}$ in $MS$; $l = l + 1$;
     }
     else
         if($t_2$ > $t_1$) then
              Insert $x_{1k}$ in $MS$; $k = k + 1$;
         else
              Insert $x_{2l}$ in $MS$; $l = l + 1$;
  }
  if($k == m_1 + 1$) then
     Append remaining vehicles on $Road_1$ to MS
  else
     Append remaining vehicles on $Road_2$ to MS

HoL Algorithm End
```

4. Continuous stream of vehicles

The optimization formulation and HoL algorithms described above are applicable only for a snapshot of real world scenario. In reality, there is continuous inflow of vehicles in AoI and hence we need to extend these algorithms to deal with it. This involves following issues:

- Identifying the snapshot of vehicles to which algorithms will be applied
- Determining how often these snapshots should be captured i.e., how often the algorithm is run

In this section, we have described two ways in which our algorithms can be tuned to address these issues.

Sporadic Approach. The above algorithms can be run sporadically on vehicle snapshots i.e., all the vehicles present in AoI. This approach makes 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 vehicle outside the AoI 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 task can be determined by imposing a lower bound on x.

Multi-Zone Approach. The drawbacks of sporadic approach is whenever a new vehicle enters the AoI: (i) *computational overhead:* it reconsiders all the vehicles (except those who passed through the merge region) from previous snapshot for determining the solution and (ii) *stability concern:* reconsidering the vehicles which are nearer to merge region might pose a threat to system stability. To overcome these drawbacks, in this approach the AoI region is divided into three zones as shown in the Figure 2. In this approach, the snapshot comprises of all the vehicles present in zone 2. Initially solution is computed for a snapshot. When vehicles from zone 3 enter zone 2 or after time δ, whichever is minimum, the algorithm takes the next snapshot and computes the solution. Note that the vehicles which enter zone 1 are left undisturbed, as these vehicles are very close to the merge region and have very little flexibility to adapt any changes to their profile. The parameter δ can be computed using the radii of the zones and V_{max}. Thus solution can be computed sporadically to deal with the continuous stream of vehicles. The work is in progress to formally characterize these zones.

5. Simulations and Observations

This section describes Matlab-based evaluations of optimization formulation and C++ simulation of HoL approach and observations made from these.

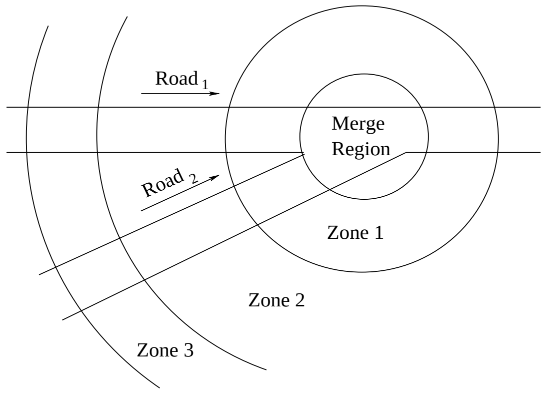

Figure 2. Region Partitioning

For simplicity, we considered a 2-road intersection problem where each road is having five vehicles. For modeling optimization formulation in MATLAB, we used Optimization Toolbox (function *fmincon*). Various parameters of the system were set to following values while performing experiments:
$A_{max} = 4m/s^2, A_{min} = -4m/s^2$,
$V_{max} = 27m/s, V_{min} = 0m/s$, $S = 5m$

Input: The vehicle profiles at time $t = 0$ (which system takes as input) is shown in Table 2. For instance, entries in the first row of the table represent: the initial velocity of vehicles x_{11} and x_{12} are set to $20m/s$ and $22m/s$ and their distances from the intersection region are set to $55m$ and $40m$ respectively. The acceleration of all the vehicles are assumed to be zero initially i.e. the vehicles are moving with uniform velocity u_{ij}.

Table 2. Initial vehicle profiles (i.e., at time t=0)

Road1				**Road2**			
Id	**u(m/s)**	**S(m)**	**Acc(m/s^2)**	**Id**	**u(m/s)**	**S(m)**	**Acc(m/s^2)**
1	20	55	0	1	22	40	0
2	22	62	0	2	20	60	0
3	21	69	0	3	25	73	0
4	25	84	0	4	22	80	0
5	23	91	0	5	21	87	0

Output: The algorithms came up with the time (i.e., Merge Sequence order) at which each vehicle is allowed to enter the intersection region which is depicted along with acceleration of the vehicles in Table 3.

As we can see, the average latency obtained using both approaches are quite comparable. Also the merge sequence order was observed to be the same in

Table 3. Simulation results showing the DTTI of all vehicles and the merge sequence

Optimization Formulation				**Head of Lane**			
Road Id	**Veh Id**	**Time(s)**	**Acc(m/s^2)**	**Road Id**	**Veh Id**	**Time(s)**	**Acc(m/s^2)**
2	1	1.63	3.06	2	1	1.63	3.06
1	1	2.34	2.99	1	1	2.34	2.99
1	2	2.53	1.98	1	2	2.53	1.98
2	2	2.72	1.54	2	2	2.71	1.55
2	3	2.92	-0.01	2	3	2.92	0.00
1	3	3.12	0.70	1	3	3.12	0.72
1	4	3.34	0.10	1	4	3.33	0.12
2	4	3.54	0.35	2	4	3.53	0.37
1	5	3.75	0.67	1	5	3.75	0.69
2	5	4.10	0.10	2	5	3.94	0.55
		29.99				**29.81**	

both the approaches. The results from the optimal approach are slightly inferior than those from the HoL approach. These inconsistencies can be attributed to the fact that function $fmincon$ is a derivative-based search algorithm and it does not guarantee a global optimum [Coleman et al.,].

Figure 3 and 4 show the same results when plotted as graphs. 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).

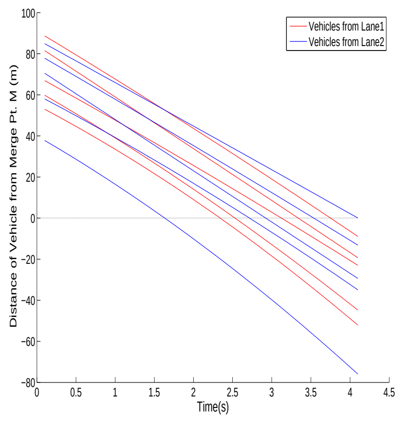

Figure 3. Optimization formulation results

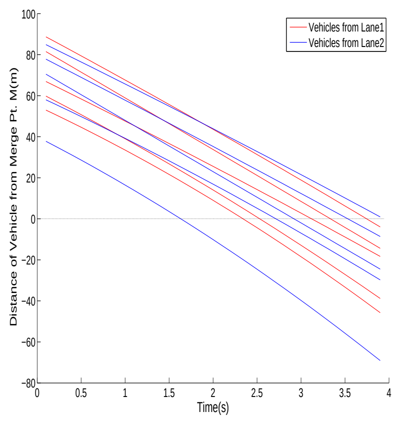

Figure 4. HoL results

It can be observed 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). It should 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.

While conducting experiments, it was observed that optimization formulation approach is computationally intensive compared to HoL. This observation can be attributed to the way optimization formulation functions i.e., it considers several possible combinations by considering all vehicles on each road at a time for determining the merge sequence whereas HoL considers only head of lane vehicles at a time for determining the same.

6. Related Work

The merge control application with inter-vehicle communication is also studied in [Uno and Tsugawa, 1999]. It uses the concept of *virtual vehicle* that is used to map vehicles on 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 [Bruns and Munch, 2006] where initially computed suboptimal velocity profiles of vehicles gets refined to optimal profiles as the vehicles approach the zone nearer to intersection region. 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.
In [Dresner and Stone, 2005], a reservation based multi-agent (reservation manager and driver agent) approach is proposed for designing the intersection control system. The driver agents "call ahead" to the intersection manager and request space-time in the intersection. The intersection manager then determines whether or not these requests can be met. If the request is met then the driver agent records the parameters of the request (the reservation) and attempts to meet them, else it sends the request again by adapting vehicle's velocity. This work comes close to ours. We believe the main drawback of this approach is the process of repeated requests by the driver agent when its initial request is not met. The intersection manager should be more smart to make use of all the vehicles' information available with it and suggest or block an alternate space-time in the intersection instead of rejecting the request and wait for that driver agent to make another request.

7. 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. We also presented HoL approach which is less computationally intensive and whose performance (merge sequence, DTTI of vehicles) is comparable to that of optimization approach. The observations from simulations carried out confirm these things.

We are working on several possible extensions to the research described here. First, extending the HoL approach for n-road intersection scenario considering the effect on previous vehicles while determining the cost associated with each vehicles (cascading effect described in the paper). Second, decentralizing the proposed approaches in which vehicles communicate with each other and resolve any conflicts among themselves without the help of any centralized controller. The real-time communication protocols for the decentralized approach are being studied. Third, fine tune our approaches to be able to consider several factors driven by real-world constraints such as giving preference to vehicles on a particular road, particular vehicles (say ambulance), angle of intersection of roads, etc. Fourth, augmenting the existing mechanisms to deal with a mix of automated vehicles and human driven vehicles. Lastly, provide real-time support for the system and demonstrate the concepts on robotic vehicular platforms built in our lab.

References

Bruns, Tornsten and Munch, Eckehard (2006). Intersection management as self-organisation of mechatronic systems. In *Proceedings of the 6th International Heinz Nixdorf Symposium on New Trends in Parallel and Distributed Computing*, Paderborn, Germany.

Coleman, Thomas, Branch, Mary Ann, and Grace, Andrew. Optimization toolbox for use with matlab user's guide version 2.

Dresner, Kurt and Stone, Peter (2005). Multiagent traffic management: An improved intersection control mechanism. In Dignum, Frank, Dignum, Virginia, Koenig, Sven, Kraus, Sarit, Singh, Munindar P., and Wooldridge, Michael, editors, *The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, New York, NY. ACM Press.

Raravi, Gurulingesh, Sharma, Neera, Ramamritham, Krithi, and Malewar, Sachitanand (2006). Efficient real-time support for automotive applications: A case study. In *Proceedings of the 12th IEEE International Conference on RTCSA*, pages 335–341, Sydney, Australia.

Uno, A. Sakaguchi, T. and Tsugawa, S. (1999). A merging control algorithm based on inter-vehicle communication. In *IEEE International Conference on Intelligent Transportation Systems*, pages 783–787, Tokyo, Japan.

Vahidi, Ardalan and Eskandarian, Azim (2003). Research advances in intelligent collision avoidance and adaptive cruise control. *IEEE Transactions on Intelligent Transportation Systems,*, 4:143–153.