Automotive Intersections

INSERT PICTURE OF CRAZY INTERSECTION HERE

Autonomous Intersections

- Future roadways face congestion issues due to increasing volume
- Autonomous cars will become much more prevalent in future
- GOAL: Model an intelligent, autonomous vehicle-based intersection system with a centralized controller intead of P2P protocol

Original Vehicle Model

Originally, we assumed an intersection environment that contained both human and autonomously controlled vehicles where:

- Human controlled vehicles respond only to $START$ and $STOP$ commands by the controller
- Autonomous vehicles repond to $START$, $STOP$, $CHNG$, and $SLWDN$
- Controller analyzes all vehicles in system and sends out commands to optimize throughput
- The intersection is a hybrid system containing both discrete and continuous components

Controller Knowledge

Controller has continuous knowledge of certain vehicle properties at all times within the intersection environment due to *embedded wireless communication devices* in each vehicle continously sending:

- Current Speed
- Route Planned
- Entrance Lane
- Location

Deviations from Original Plan

Changes in Intersection System Model:

- Shift of focus from optimization towards correct modelling
- $a(t) = 0$
- Changes in velocity can occur instantaneously
- System contains autonomous vehicles only
- Vehicle reaction to controller command is instantaneous
- Intersection turns have no effect on velocity

Reasons for Changes

- Optimization of the system may be intractable for application
- Human controlled vehicles would be a rarity in the described system
- Constant velocity model reasonable within limited intersection distances
- Limitations with dynamic actor instantiation in Ptolemy modelling
- Learning curve involved with modelling tool (Ptolemy)

Modelling Objectives

Model Objectives

Let c denote the event in which a collision event has occured at time t, and let e denote the event in which a car, v, has entered a route within the intersection.

- Safety: $\forall t : \neg Gc$
- Fairness: $\forall v : Fe$

Modelling Tool

Modelling Tool: Ptolemy II Version 8.0.1
INSERT PIC HERE

Assumptions

Vehicle Properties

Properties exhibited by each vehicle in the model:

- Instantaneous change in velocity
- Constant velocity within intersection
- Velocity (mph): $V \to V \in [20, 40]$
- Route intention constant throughout intersection traversal
- Randomized vehicle entry times
- Receives velocity change or delayed start time from controller

Intersection Model

INSERT PIC HERE

Modelling Continuous Vehicle Flow

Ptolemy limitations prevent dynamic instantiation of actors.

Multi-Instance Composite Actor

- Vehicle instances accessed by ID encoding
- Create N vehicle Modal Model instances
- Generate random:
  - Intersection entry time: $1 - 45$ secs
  - Initial velocity (m/s): $V_0 \in [20, 40]$
  - Direction: N, S, E, W
  - Route: $L, S1, S2, R$
- Implement as many instances of the car model as desired
- Interacts with Python-based controller
- Car can "re-enter" system only after it exits (re-use vehicles vs. generating new vehicles)

Vehicle Modelling

INSERT PIC HERE

Vehicle FSM Actor

States

- RUN
- $IDLE$
- $ENTER$
- $BOOK$
- GO
- $WAIT$

Vehicle FSM Actor

INSERT PIC HERE

ENTER State

ENTER State

Abstraction:

- Equivalent to pre-conflict zone straightaway
- Initial velocity used to calculate time until intersection
- Stops at intersection if instructions not received from controller

Guard(s);

- $t_{GLOBAL} > t_{ENTER}$

ENTER State

INSERT PIC HERE

GO State

GO State

Abstraction:

- Equivalent to the approach through the intersection
- Tracks when the vehicle leaves the system
- Stops at intersection if instructions not received from controller

Guard(s);

- $t_{GLOBAL} > t_{STRAIGHTAWAY}$

GO State

INSERT PIC HERE

Controller Properties

Controller

- Lack of multiport output (serial communication)
- Delayed feedback to vehicle
  - Prevent a zeno system
  - Models communication delay
- Receives requests and sends commands to system vehicles
- Inputs to controller: $velocity$, $booking$, $entertime$
- Python script actor

Ptolemy Model

INSERT PIC HERE

Algorithm Overview

Initialize a list of conflict zones and their distances from the starting points of the intersections along the curved paths.
At every time step: current = [] t = ContinuousDirector.currentTime add all the tokens received from Vehicle modal model to current for each car in current: if car needs a booking: find max velocity in [currentVel - 10, currentVel + 10] which ensures safety (no collisions) if max velocity found: broadcast(returnvelocity = max velocity, delay = 0, id = car.id) if no such velocity found: set returnvelocity = currentVel + 10
if none of the set delay = end of latest booked time interval for all conflict zones broadcast(returnvelocity, delay, id = car.id)

Algorithm Walkthrough

Show Ptolemy Model

Top 4 Challenges

- Continuous car flow given static number of vehicles
- Delivering controller commands to specified car instance
- Continuous time solver time step evaluation issues
- Hybrid system causality issues

Summary

Project Conclusions

Achieved:

- Functional hybrid model
- Continuous vehicle flow with static vehicle set
- Centralized responsive controller
- Rigorous intersection path definitions
- Multiplexed communication to vehicles

Further Work:

- Further optimize controller response algorithm
- Introduce vehicle acceleration into the model
- Add support for human cars