The MIT – Cornell Collision and Why it Happened

Luke Fletcher, Seth Teller, Edwin Olson
David Moore, Yoshiaki Kuwata
Jonathan How, John Leonard
Team MIT
Massachusetts Institute of Technology
Cambridge, MA 02139
lukesf@mit.edu

Isaac Miller, Mark Campbell,
Dan Huttenlocher, Aaron Nathan,
Frank-Robert Kline
Team Cornell
Cornell University
Ithaca, NY 14853
itm2@cornell.edu

Abstract

Mid-way through the 2007 DARPA Urban Challenge, MIT's autonomous Land Rover LR3 'Talos' and Team Cornell's autonomous Chevrolet Tahoe 'Skynet' collided in a low-speed accident, one of the first well-documented collisions between two full-size autonomous vehicles. This collaborative study between MIT and Cornell examines the root causes of the collision, which are identified in both teams' system designs. Systems-level descriptions of both autonomous vehicles are given, and additional detail is provided on sub-systems and algorithms implicated in the collision. A brief summary of robot–robot interactions during the race is presented, followed by an in-depth analysis of both robots' behaviors leading up to and during the Skynet–Talos collision. Data logs from the vehicles are used to show the gulf between autonomous and human-driven vehicle behavior at low speeds and close proximities. Contributing factors are shown to be: (1) difficulties in sensor data association leading to phantom obstacles and an inability to detect slow moving vehicles, (2) failure to anticipate vehicle intent, and (3) an over emphasis on lane constraints versus vehicle proximity in motion planning. Eye contact between human road users is a crucial communications channel for slow-moving close encounters between vehicles. Inter-vehicle communication may play a similar role for autonomous vehicles; however, there are availability and denial-of-service issues to be addressed.

1 Introduction

On November 3rd, 2007, the Defense Advanced Research Projects Agency (DARPA) Urban Challenge Event (UCE) was held in Victorville, California. This event set loose, simultaneously, for the first time, 11 full-size autonomous vehicles on a closed course. The aim of the contest was to test the vehicles' ability to drive between checkpoints while obeying the California traffic code. This required exhibiting behaviors including lane keeping, intersection precedence, queuing, parking, merging and passing.

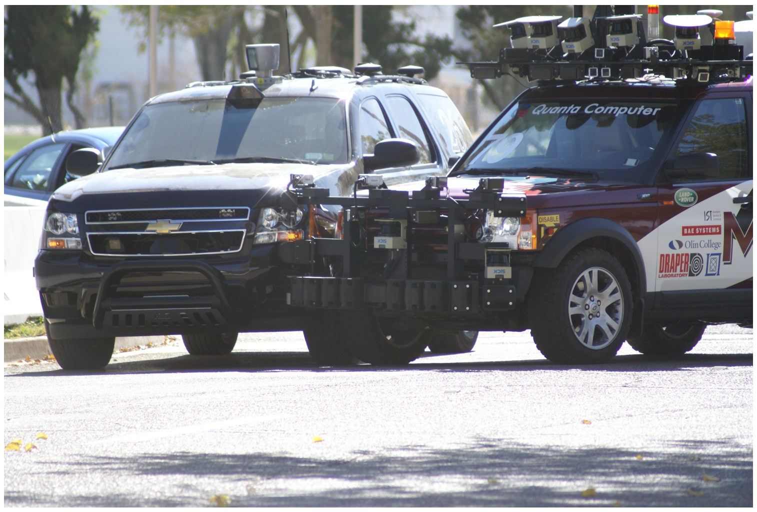


Figure 1: The collision. *(left): Skynet. (right): Talos.*

On the whole, the robots drove predictably and safely through the urban road network. None of the robots stressed the (understandably) conservative safety measures taken by DARPA. There were, however, a number of low-speed incidents during the challenge. This paper takes an in-depth look at one incident, the collision between Team Cornell's vehicle 'Skynet' and MIT's 'Talos'. This paper scrutinizes why the collision occurred and attempts to draw some lessons applicable to the future development of autonomous vehicles.

The UCE was held on a closed course within the decommissioned George Air-force base. The course was predominantly the street network of the residential zone of the former base with several graded dirt roads added for the competition. The contest was cast as a race against time to complete 3 missions. The missions were different for each team but were designed to require each team to drive 60 miles to finish the race. Penalties for erroneous or dangerous behavior were converted into time penalties. DARPA provided all teams with a single Route Network Definition File (RNDF) 24 hours before the race. The RNDF is very similar to a digital street map used by an in-car GPS navigation system. The file defined the road positions, number of lanes, intersections, and even parking-space locations in GPS coordinates. A plot of the route network for the race is shown in Figure 2. On the day of the race, each team was provided with a second unique file called a Mission Definition File (MDF). This file consisted solely of list of checkpoints within the RNDF which the vehicle was required to cross.

To mark progress through each mission, DARPA arranged the checkpoints in the mission files to require the autonomous vehicle to return to complete a lap of the oval shaped "Main Circuit" (visible in bottom left corner of Figure 2) at the end of each sub-mission. Each mission was subdivided into 6 or 7 'sub-missions'. At the end of each mission, the vehicles returned to the finishing area, where the team could recover and reposition the vehicle for the next mission. Most roads were paved with a single lane in each direction, similar to an urban road. Several roads had two lanes of traffic in each direction, like an arterial road or highway. One road, in the southeastern corner of the network, was a raised dirt road constructed especially for the event.

All 11 qualifying robots were allowed to interact in the UCE course simultaneously, with additional traffic supplied by human-driven Ford Tauruses. To prevent serious crashes during the competition, all autonomous vehicles were followed by an assigned DARPA chase vehicle. The chase-vehicle driver supervised the robot and could 'Pause' or, in extreme cases, 'Disable' the robot via radio link. 'Paused' robots could then be 'Un-Paused' to continue a mission when safe. 'Disabling' a vehicle would kill the engine, requiring the vehicle's team to recover it.

The qualifiers and the race provided ample opportunity for damage to the robots on parked cars, concrete barriers, DARPA traffic vehicles and buildings. The fact that the two vehicles were not damaged, other than minor scrapes in the collision, despite hours of driving emphasizes the fact that the circumstances leading to the collision were the product of confounding assumptions across the two vehicle architectures. The robots negotiated many similarly complex situations successfully.

This paper begins with a brief summary in Section 2 of the robot–robot interactions during the 6-hour race. Then, to aid in the collision analysis, summaries of the MIT and Cornell vehicle software architectures are given in Sections 3 and 4 respectively. Section 5 describes the *Skynet–Talos* collision in detail, before branching in Sections 7 and 6 to provide detailed accounts of the robots' software state during the incident. The apparent causes of the incidents are studied here to shed light on the deeper design issues involved. In Section 8, we draw together the insights from the software architecture analysis to summarize the common themes, the lessons learned, and the impediments to using these robots on the real urban roads.

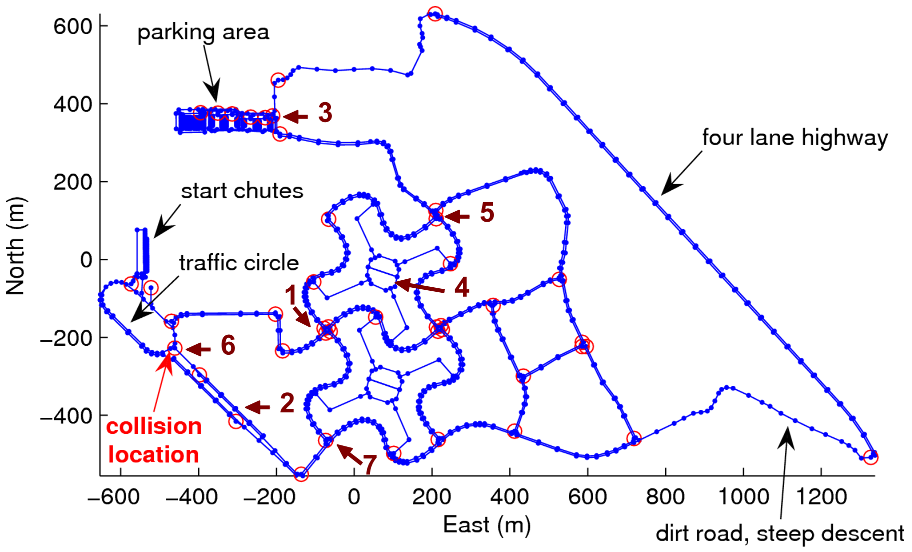


Figure 2: The UCE road network. Waypoints are designated as blue dots, with traversable lanes and zone boundaries represented as blue lines. Stop lines are designated as red circles. The *Skynet–Talos* collision happened entering the Main Circuit on the bottom left.

2 Chronology of Robot–Robot interactions

The following table is a list of robot–robot collisions or close calls during the UCE. The list has been compiled from the race day web-cast and vehicle data logs. The locations of the incidents are marked in Figure 2.

Time (Approx)	Location	Description	Reference
1h00m	Utah and Washington	Cornell's *Skynet* passing with IVS' *XAV-250* and Ben Franklin Racing Team's *Ben* oncoming	Section 2.1
1h30m	George Boulevard	*Ben* and Team UCF's *Knight Rider*	Section 2.2
2h00m	North Nevada and Red Zone	CarOLO's *Caroline* turns across MIT's *Talos*	Section 2.3
3h00m	White Zone	*Caroline* and *Talos* collide.	Section 2.4
4h00m	Carolina Avenue and Texas Avenue	*Talos* swerves to avoid Victor Tango's *Odin*	Section 2.5
4h30m	George Boulevard and Main Circuit	*Skynet* and *Talos* collide	Section 5
5h20m	Utah and Montana	*Talos* turns across *Ben*	Section 2.6

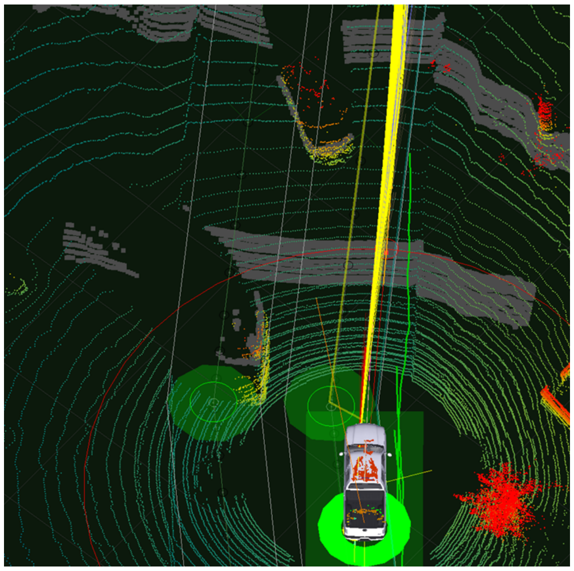
Teams with vehicles actively involved the incidents (CarOLO, IVS and Ben Franklin Racing Team) were invited to co-author/comment on the interactions. Any comments received are included in the descriptions that follow.
A full discussion of the *Skynet*– *Talos* collision is given Section 5.

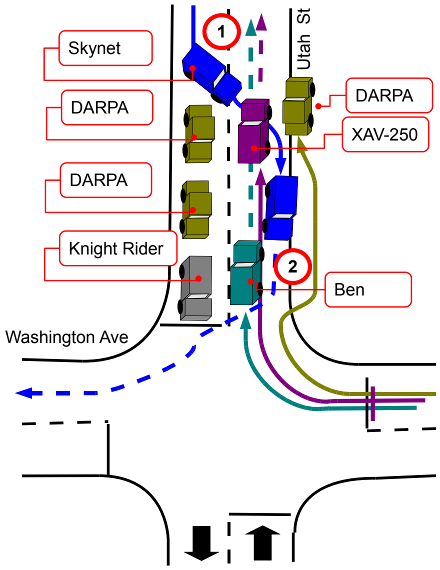
Diagrams have been drawn describing each incident. In the drawings, a solid line shows the path of the vehicle, and a dashed line shows the intended/future path of the vehicle. A lateral line across the path indicates that the vehicle came to a stop in this location. DARPA vehicles are driven by DARPA personnel in the roles of either traffic or chase vehicles.

Videos of the log visualization for incidents involving *Talos* can be found in Section 9).

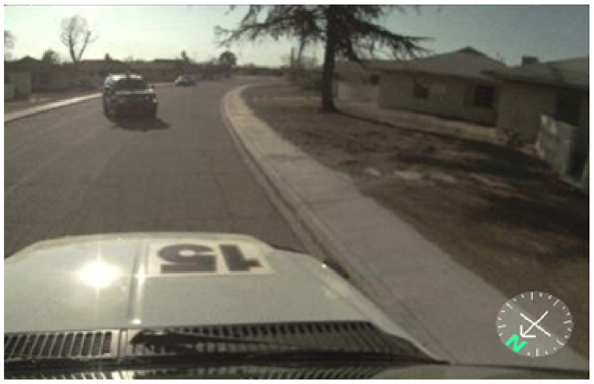
2.1 *Skynet* passing with *XAV-250* and *Ben* oncoming at Utah and Washington

The first near-miss occurred at the intersection of Utah and Washington. *Knight Rider* was at the intersection. *Skynet* pulled up behind a traffic vehicle, which was queued behind *Knight Rider's* chase vehicle (the chase vehicle is queued behind *Knight Rider*). The relative positions of the


(a)



(b)


(c)

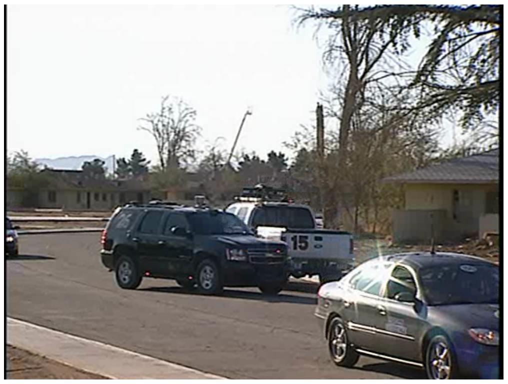
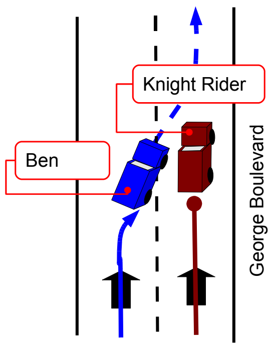

(d)

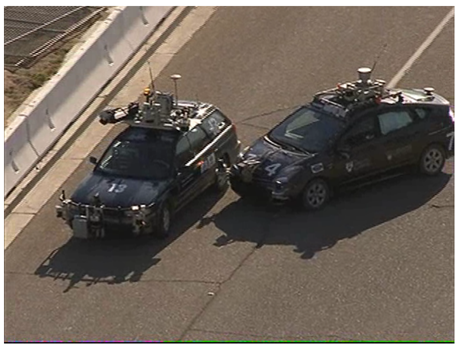
Figure 3: After queuing behind the stationary *Knight Rider*, *Skynet* passes. (a) Visualization of *XAV-250* log when approaching *Skynet* (Image courtesy of Team IVS).(b) Diagram of incident. *(1):* Vehicle positions when *XAV-250* approaches. *(2):* Vehicle positions when *Ben* approaches. (c) *XAV-250* Camera view (Image courtesy of Team IVS). (d) *Skynet*$_{(26)}$ once *XAV-250*$_{(15)}$ had passed.

vehicles are shown in Figure 3(b). *Knight Rider* was making no apparent progress through the intersection, so after waiting, *Skynet* elected to pass. *Skynet* was behind three cars, which put it beyond the safety zone in which passing was prohibited. (DARPA, 2007). The rules also stated that vehicles should enter a traffic-jam mode after a prolonged lack of progress at an intersection. *Skynet* began to pass. Shortly into the maneuver, the Intelligent Vehicle Systems vehicle *XAV-250* turned right from Washington onto Utah and into the on-coming path of *Skynet*. *Skynet* and *XAV-250* were Paused. *XAV-250* was Un-paused and permitted to drive past *Skynet*, clearing the area. *Skynet* was then also permitted to continue. *Skynet* determined that it could not get back into the correct lane and was too near the intersection, so it pulled over to the curb side of the lane and waited. Next *Ben* also turned onto Utah from Washington, and again, was on-coming to *Skynet*. *Skynet* and *Ben* were Paused . *Ben* was Un-paused and permitted to drive past. Interestingly, *Ben*'s chase vehicle drove onto the curb around to the right to pass the *Skynet* vehicle. This provides an example of the assessment made by a human driver in this scenario. Faced with *Skynet* in the on-coming lane, the chase vehicle driver elected to drive far right onto the curb to accommodate the potential behavior of the *Skynet* vehicle. The passage of *Ben* shows that mounting the curb was not necessary to physically pass the vehicle. Given a clear intersection, the *Skynet* vehicle was able to negotiate the intersection and continue the mission. A more detailed account of this event from *Skynet*'s point of view is given in (Miller et al., 2008).

2.2 *Ben* and *Knight Rider* on George Boulevard



(a)

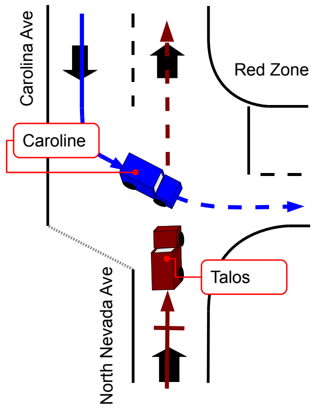


(b)

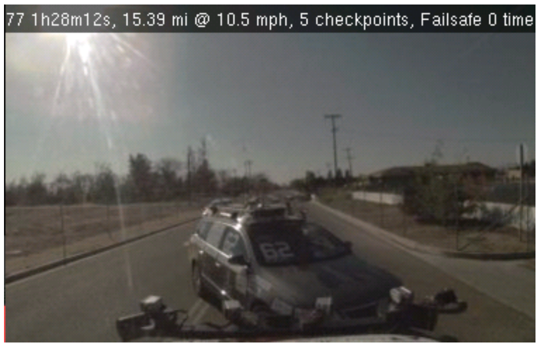
Figure 4: (a) Diagram of incident. (b) *Knight Rider*$_{(13)}$ and *Ben*$_{(74)}$ near miss.

Figure 4 shows a near-miss featured in the webcast in which *Ben* appeared to be merging into *Knight Rider* on George Boulevard. In this case, *Ben* had been following *Knight Rider*. *Ben* had been traveling faster than *Knight Rider* so DARPA decided to Pause *Knight Rider* once the vehicles were on George Boulevard, a dual lane road, to permit *Ben* to pass. With *Knight Rider* stopped in the right lane, DARPA expected *Ben* to continue in the left lane and then merge into the right lane after passing *Knight Rider*. At the far end of George Boulevard, the vehicles were required to be in the right lane. Because it involved a lane change at the start of a dual-lane road and the stopped vehicle was in the destination lane, the scenario was different from standard passing maneuvers and hence was not handled in *Ben's* software. Consequently *Ben* performed the lane change without accounting for *Knight Rider*. *Ben* was Paused and stopped in time to prevent a collision.

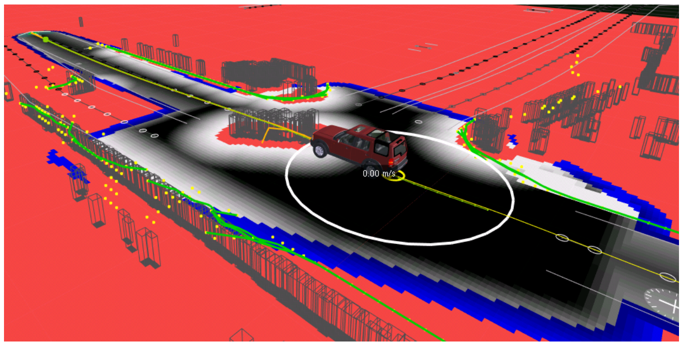
2.3 *Caroline* and *Talos* at North Nevada and Red Zone



(a)


77 1h28m12s, 15.39 mi @ 10.5 mph, 5 checkpoints, Failsafe 0 time

(b)



(c)

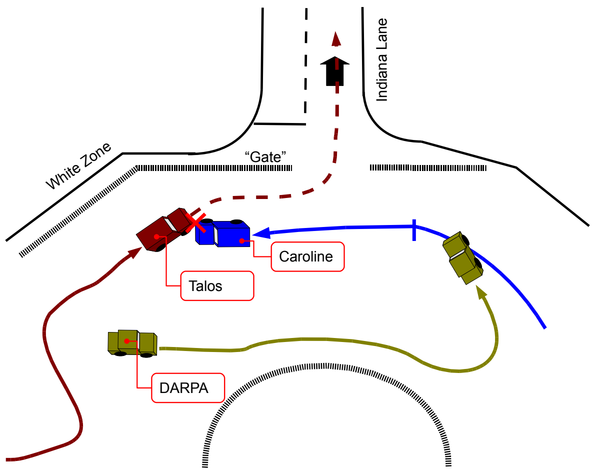
Figure 5: (a) Diagram of incident. (b) *Talos'* view of the final pose *Caroline*-*Talos* turning near-miss. (c) Visualization from *Talos'* log.

Figure 5 shows the first of two close encounters between *Caroline* and *Talos*. The figure shows the diagram of the incident and how it appeared in the *Talos* software. In this incident *Talos* was driving straight down North Nevada. *Caroline* was approaching in the oncoming direction down Carolina Avenue, then turns left into the Red Zone across the path of *Talos*.

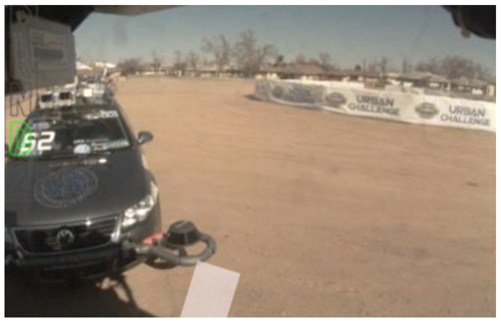
Talos detects the moving object of *Caroline* and finds the closest intersection exit to project the assumed trajectory. *Talos'* intended motion plans are then severed by *Caroline's* predicted trajectory, so the vehicle commences an emergency stop. DARPA Pauses both vehicles. A full account of this event from *Talos'* view is given in (Leonard et al., 2008).

2.4 *Caroline* and *Talos* in White Zone

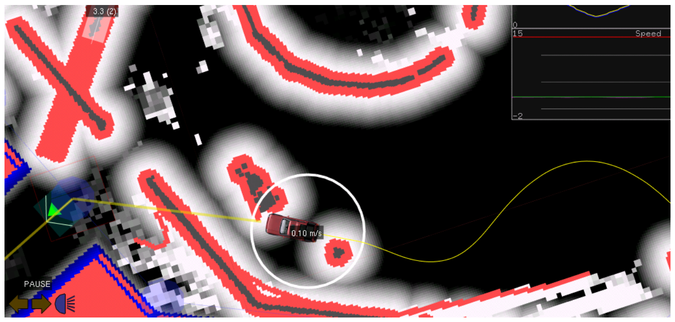
The second incident between *Caroline* and *Talos* ended in a collision. The *Caroline* vehicle was retired from the race shortly after this event. Figure 6 shows the diagram of the incident and collision between *Caroline* and *Talos* in the White Zone. *Caroline* was near the Indiana Lane exit of the White Zone. *Talos* entered the Kentucky Lane entrance to the White Zone and was en-route to the Indiana Lane exit. Initially, *Talos* planned a route around *Caroline's* chase vehicle to



(a)



(b)



(c)

Figure 6: (a) Diagram of incident. (b) Final pose of *Caroline* and *Talos* collision from *Talos'* front right camera. (c) Visualization from *Talos'* log.

get to the Zone exit on the left. *Caroline's* chase vehicle then drove away from *Talos*. *Talos* then replanned a more direct route to the left, to the zone exit. As *Talos* drove toward the zone exit, *Talos* also approached *Caroline*. Initially *Caroline* was stationary, then *Caroline* drove slowly forward toward *Talos*. *Talos*, with the zone fence to the left and what it perceived as a static obstacle (that was actually *Caroline*) to the right, attempted to negotiate a path in between. *Caroline* advances toward *Talos*. *Talos* keeps adjusting its planned path to drive around to the left of what appears to the *Talos* software as a stationary object. Just before the collision *Talos*' motion plans are severed causing a "planner emergency stop". Due to *Talos'* momentum and *Caroline's* forward movement, the braking failed to prevent physical contact. DARPA then Paused the vehicles. A detailed account of this chain of events from *Talos'* view is given in (Leonard et al., 2008).

2.5 *Odin* and *Talos* at Carolina and Texas

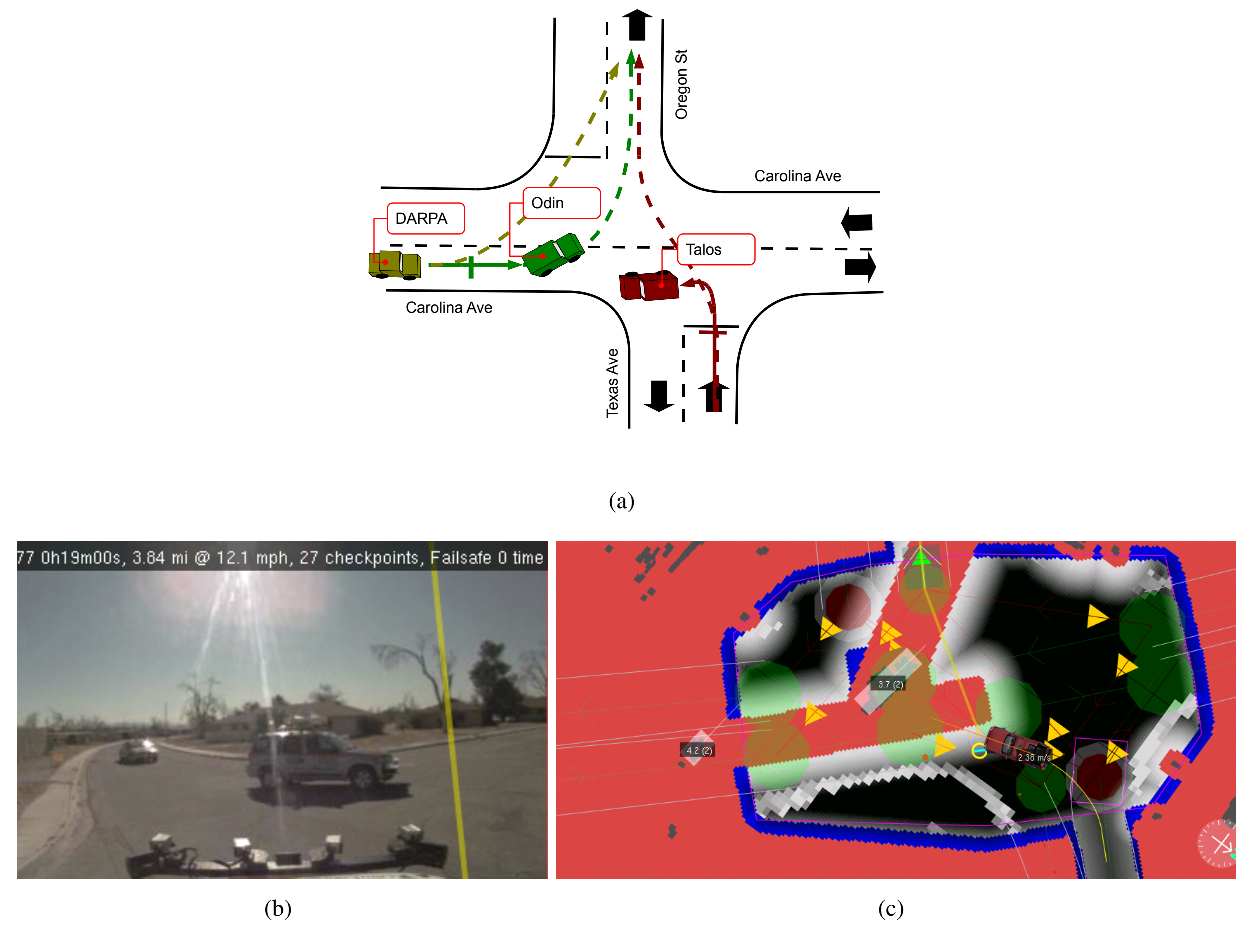
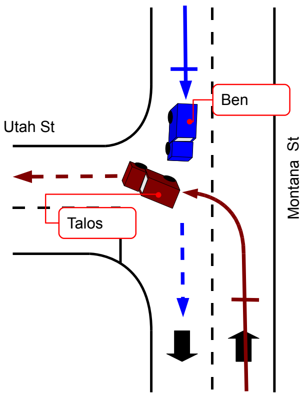


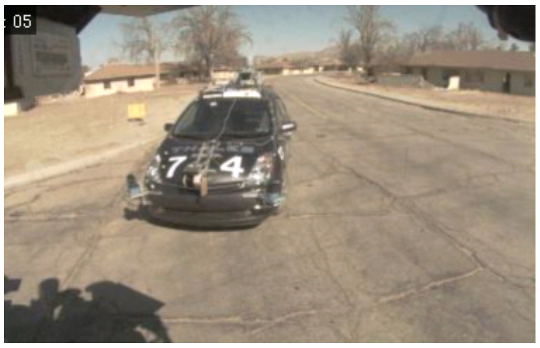
Figure 7: (a) Diagram of incident. (b) View from *Talos'* front camera. (c) *Talos'* log visualization. *Odin* is turns right. *Talos* brakes and turns hard left to avoid *Odin's* projected motion direction.

This incident featured a close call negotiated by the robots without intervention from DARPA. Figure 7 shows the diagram and view from the *Talos* log. *Talos* arrives at a stop line at the intersection of Carolina and Texas. *Talos* is intending to go from Oregon to Texas (from bottom to top in Figure 7(a)). *Talos* yields to *Odin* approaching. *Odin* arrives at the intersection intending to turn left into Texas Avenue. *Odin* comes to a stop entering the intersection. *Talos* detects the time to contact for approaching vehicles has gone to infinity so proceeds across the intersection. *Odin* also proceeds from Carolina Avenue into Texas Avenue. *Odin*, much quicker off the mark, is ahead of *Talos*. *Talos* reacts to *Odin* approaching by braking and replanning an evasive maneuver, turning hard to the left. *Odin* and *Odin's* chase vehicle complete the turn and clear the intersection. *Talos* then resumes course down Texas Avenue behind the vehicles.

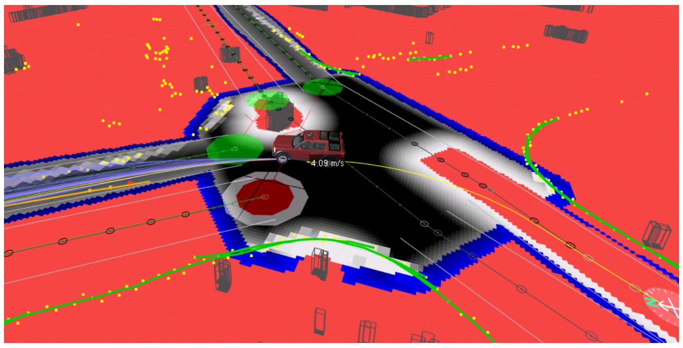
2.6 *Ben* and *Talos* at Utah and Montana



(a)



(b)



(c)

Figure 8: (a) Diagram of incident. (b) View from *Talos'* right front camera. *Talos'* view of the *Little Ben-Talos* turning near-miss. *Talos* yields to the velocity track of oncoming $Ben_{(74)}$. *Ben* comes to a stop at the intersection. *Talos* begins motion. *Ben* begins to go through the intersection. *Talos* sees *Ben* as a creeping "static obstacle" and continues. *Talos* completes the turn. *Ben* stops. (c) *Talos'* log visualization. *Ben* approaching on the right of *Talos*.

The final incident, a close call, is illustrated in Figure 8. Log data shows *Talos* turning left from Montana onto Utah. *Talos* arrives at the intersection, and yields to oncoming traffic. *Ben* is approaching, so *Talos* remains stopped. At the intersection *Ben* also comes to a stop. Again, *Talos* detects that the time to contact for approaching vehicles has now gone to infinity, so commences the left-hand turn. As *Talos* crosses in front of *Ben*, *Ben* then also enters the intersection. At this point, *Ben* is quite far to the right of *Talos*, so *Talos'* forward path collision checking is not altered by the vehicle approaching to the side. *Talos* exits the intersection while *Ben* comes to a stop. Once the intersection is clear, *Ben* continues the mission.

3 Team MIT's 'Talos'

This section is a summary of the *Talos* software architecture. The purpose of this section is to describe the vehicle software in sufficient detail to understand the vehicle behavior and contributing factors to the collision. A thorough description of the robot architecture is given in (Leonard et al., 2008).

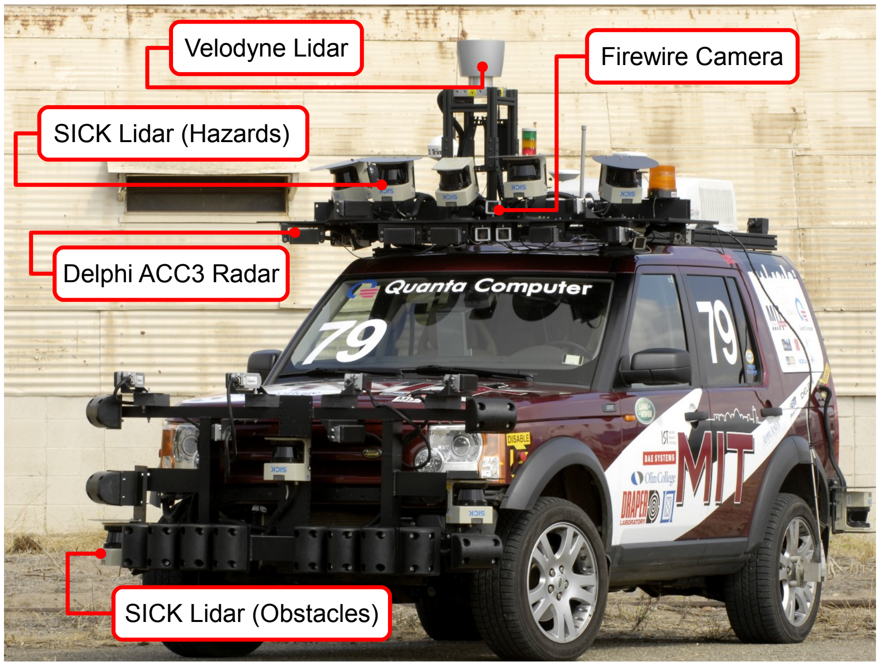


Figure 9: MIT's 'Talos', a Land Rover LR3 featuring five Point Grey FireFly cameras, 15 Dephi ACC3 radars, 12 Sick LMS-291 lidars and a Velodyne HDL-64 lidar.

Talos is a Land Rover LR3 fitted with cameras, radar and lidar sensors (shown in Figure 9). Forward, side and rear-facing cameras are used for lane marking detection. The Velodyne HDL-64 lidar is used for obstacle detection supplemented in the near-field with 7 horizontal Sick LMS-291 lidars. Five additional downward-facing Sick LMS-291 lidars are used for road-surface hazard detection including curb cuts. 15 Delphi ACC3 millimeter-wave radars are used to detect fast-approaching vehicles.

The system architecture developed for the vehicle is shown in Figure 10. All software modules run on a 40-core Quanta blade server. The general data flow of the system consists of raw sensor data processed by a set of perception software modules: the Position Estimator, Obstacle Detector, Hazard Detector, Fast [approaching] Vehicle Detector and Lane Tracker.
The Navigator process decomposed mission-level decisions into a series of short term ($1m - 60m$) motion goals and behavioral constraints. The output from the perception modules is combined with the behavioral constraints to generate a Drivability Map of the environment. The motion planning to the next short-term goal is done in the Motion Planner module with paths vetted against

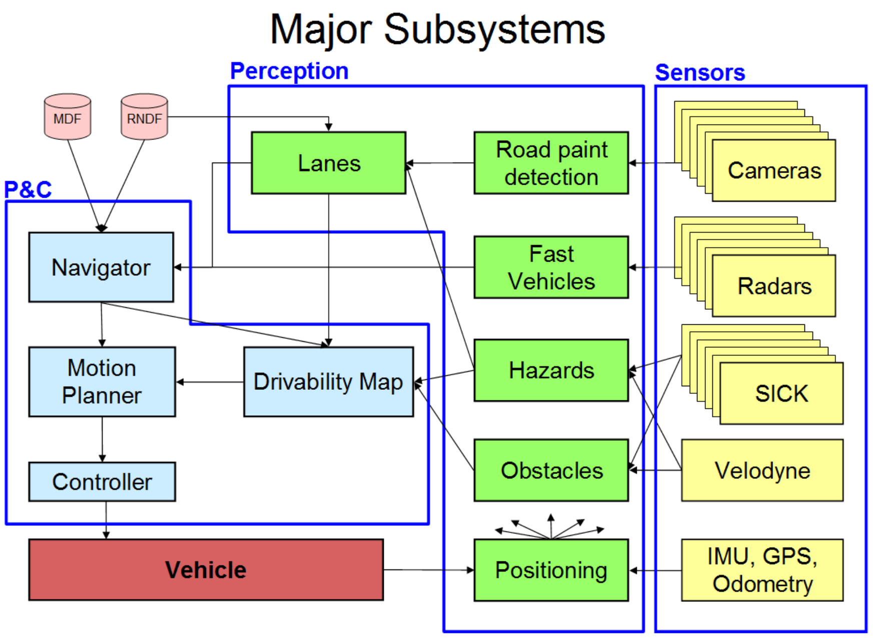


Figure 10: MIT's *Talos* system architecture.

the Drivability Map. The trajectory created by the Motion Planner is executed by the Controller module. Each module is now discussed in detail.

During the Urban Challenge, the **Navigator** tracked the mission state and developed a high-level plan to accomplish the mission based on the map (RNDF) and the mission data (MDF). The primary output was the next short-term goal to provide to the Motion Planner. As progress was made the short-term goal was moved, like a carrot in front of a donkey, to achieve the mission. In designing this subsystem, the aim was to create a resilient planning architecture that ensures that the autonomous vehicle can respond reasonably and make progress under unforeseen conditions. To prevent stalled progress, a cascade of events was triggered by a prolonged lack of progress. For example, after 10 seconds of no progress queuing behind a stationary vehicle, the Navigator would trigger the passing mode if permitted by the DARPA rules. In this mode the lane center-line constraint is relaxed, permitting the vehicle to pass. The Drivability Map would then carve out the current and oncoming lanes as drivable. After checking for oncoming traffic, the Navigator would then permit the vehicle to plan a passing trajectory around the stopped vehicle.

The **Obstacle Detector** used lidar to identify stationary and moving obstacles. Instead of attempting to classify obstacles as vehicles, the detector was designed to avoid vehicle classification using two abstract categories: "static obstacles" and moving obstacle "tracks". The output of the Obstacle Detector is a list of static obstacles, each with a location and size, as well as a list of moving obstacle "tracks", each containing position, size and an instantaneous velocity vector. The obstacle tracker integrated non-ground detections over relatively short periods of time in an accumulator. In our implementation, the tracker ran at 15Hz (matching the Velodyne frame rate). At each time step, the collection of accumulated returns were clustered into spatially-nearby "chunks". These chunks were then matched against the set of chunks from the previous time step, producing velocity estimates. Over time, the velocity estimates were fused to provide better estimates. The tracking system was able to provide velocity estimates with very low latency, increasing the safety of the

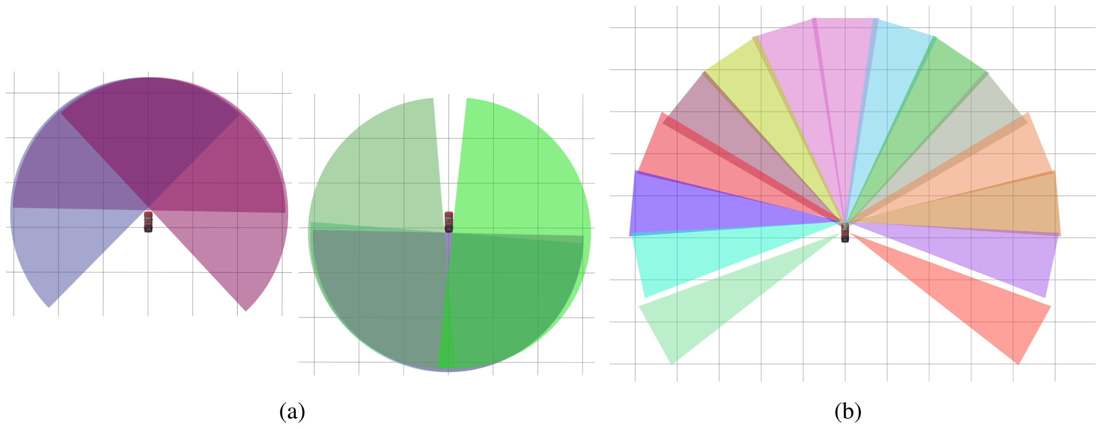


Figure 11: Sensor fields of view on 20m grid. (a) Our vehicle used a total of seven horizontally-mounted 180° planar lidars with overlapping fields of view. Front and rear Lidars are drawn separately so that the overlap can more easily be seen. For ground plane rejection two lidars were required to "see" the same obstacle to register the detection (except in the very near field). (b) 15 radars with 18°-FOV each were fanned to yield a wide (255°) total field of view.

system. The reliable detection range (with no false negatives) is about 30m, with good detections out to about 60m (but with occasional false negatives). The system was tuned to minimize false positives.

For detecting moving objects, an initial implementation simply classified any object that was approximately the size of a car, as a car. In cluttered urban settings, however, this led to many false positives. The object classification strategy was changed from a car/not-car distinction to a moving/not-moving distinction. Detecting whether something was moving was substantially easier than detecting whether an object was a car. Although this approach worked much better in general, even in cluttered environments, a new failure mode arose. In particular circumstances, stationary objects could appear to be moving, due to the changing viewpoint of our vehicle combined with aperture/occlusion effects. A high velocity threshold (3.0m/s in our implementation) was required to reduce the frequency at which stationary objects were reported to be moving.

Vehicles were also detected using the radar-based **Fast-Vehicle Detector**. The narrow 18° field of view radars were fanned to provide a 255° coverage in front of the vehicle. The raw radar detections were compensated for vehicle ego-motion then data association, and position tracking over time was used to distill the raw returns into a second set of obstacle "tracks". The instantaneous Doppler velocity measurement from the radar returns was particularly useful for detecting distant but fast-approaching vehicles. The information was used explicitly by the Navigator module to determine when it was safe to enter an intersection, or initiate merging and passing behaviors. Figure 11 shows the sensor coverage provided by Sick lidar and radar sensors.

The low-lying **Hazard Detector** used downward-looking planar Lidars mounted on the roof to assess the drivability of the road ahead and to detect curb cuts. The module consists of two parts: a hazard map, and a road-edge detector. The "hazard map" was designed to detect hazardous road surfaces by discontinuities in the lidar data that are too small to be detected by the Obstacle Detector. High values in the hazard map are rendered as high penalty areas in the Drivability Map.

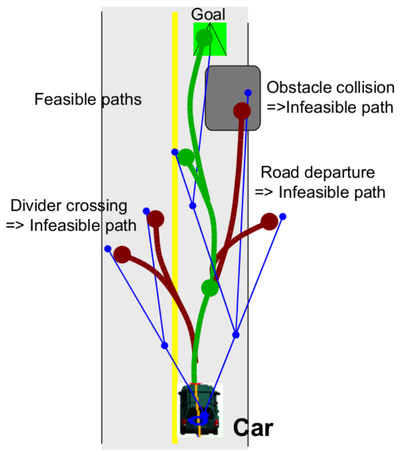
The road-edge detector looked for long strips of hazardous terrain in the hazard map. If strips of sufficiently long and straight hazardous terrain were detected, some poly lines were explicitly fitted to these regions and identified as a curb-cut or berm. These road edges were treated as obstacles: if no road paint was detected, the lane estimate would widen, and the road-edge obstacles (curbs) would guide the vehicle.

The **Lane tracker** reconciled RNDF data with lanes detected by vision and lidar. Two different road paint detectors were developed, each as a separate, stand-alone process. The first detector used a matched "Top Hat" filter scaled to the projected ground plane line width. Strong filter responses and the local gradient direction in the image were then used to fit a series of cubic Hermite splines. The second road-paint detector fitted lines to image contours bordering bright pixel regions. Both road-paint detectors produced sets of poly lines describing detected road paint in the local coordinate frame. A lane centerline estimator combined the curb and road paint detections to estimate the presence of nearby lanes. The lane centerline estimator didn't use the RNDF map to produce its estimates; it relied solely on detected features. The final stage of the lane tracking system produced the actual lane estimates by reconciling the RNDF data with the detected lane center lines. The map data was used to construct an *a-priori* estimate of the physical lanes of travel. The map estimates were then matched to the centerline estimates and a minimization problem was solved to snap the RNDF lanes to the detected lane centerlines.

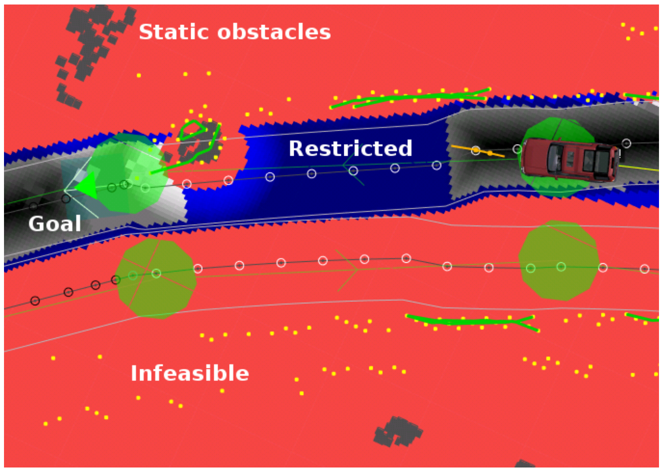
The **Drivability Map** was constructed using perceptual data filtered by the current constraints specified by the Navigator. This module provides an efficient interface to perceptual data for motion planning. Queries from the Motion Planner about future routes were validated by the Drivability Map. The Drivability Map consisted of:

- "Infeasible regions" which were no-go areas due to proximity to obstacles or just undesirable locations (such as in the path of a moving vehicle or across an empty field when the road is traversable).
- "High-cost regions" which would be avoided if possible by the motion planning and
- "Restricted regions" which were regions that could only be entered if the vehicle was able to stop in an unrestricted area further ahead.

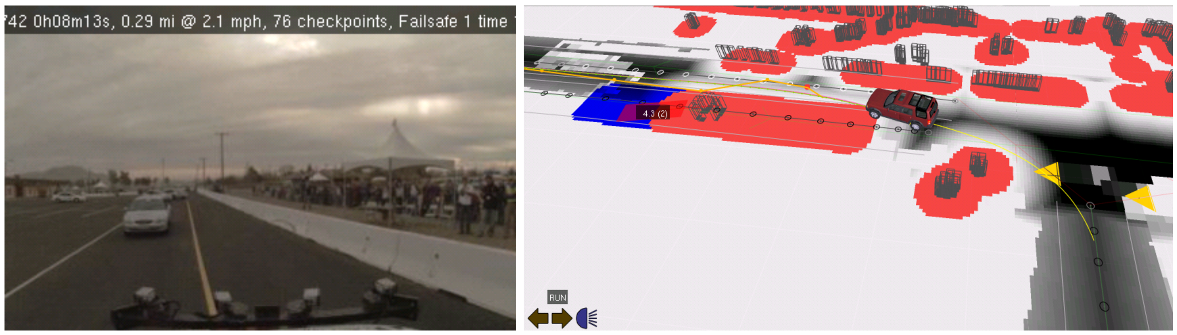
Restricted regions were used to permit minor violations of the lane boundaries if progress could be made down the road. Restricted regions were also used behind vehicles to enforce the requisite number of car lengths' stand-off distance behind a traffic vehicle. If there was enough room to pass a vehicle without crossing the lane boundary (for instance if the vehicle was parked on the side of a wide road), then Talos would traverse the Restricted region and pass the vehicle, continuing to the unrestricted region in front. If the traffic vehicle blocked the lane, then the vehicle could not enter the restricted region because there was no unrestricted place to stop. Instead, Talos would queue behind the restricted region until the traffic vehicle moved or a passing maneuver was commenced. No explicit vehicle detection was done. Instead, moving obstacles were rendered in the Drivability Map with an infeasible region projected in front of the moving obstacles in proportion to the instantaneous vehicle velocity. As shown in Figure 12(c), if the moving obstacle was in a lane the infeasible region was projected along the lane direction. If the moving obstacle was in a zone (where there was no obvious convention for the intended direction) the region was projected in the velocity direction only. In an intersection the obstacle velocity direction was compared with



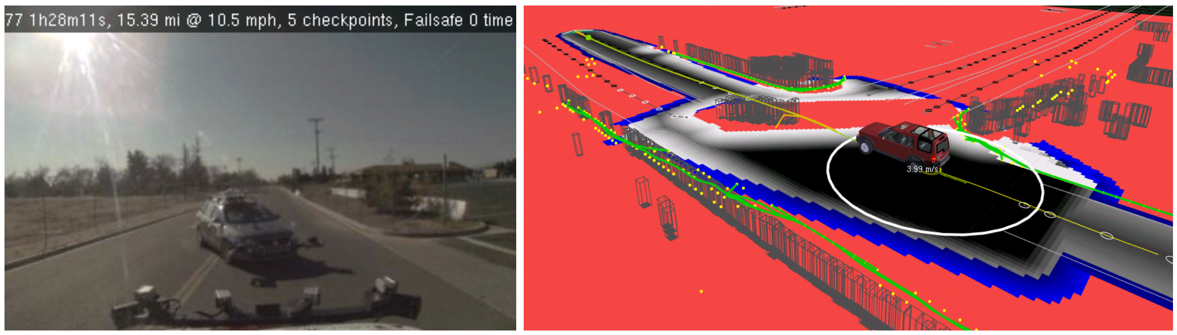
(a)



(b)


742 0h08m13s, 0.29 mi @ 2.1 mph, 76 checkpoints, Failsafe 1 time

(c)


77 1h28m11s, 15.39 mi @ 10.5 mph, 5 checkpoints, Failsafe 0 time

(d)

Figure 12: (a) RRT Motion planning. Each leaf on the tree represents a stopping location. (b) Drivability map explanation. *White arrow on green background:* Short-term goal location. *Red:* Infeasible regions are off-limits to the vehicle. *Blue:* Restricted regions may only be entered if the vehicle can stop in an unrestricted region further on. *White or Gray:* High-cost regions accessible to the vehicle. Dark areas represent low-cost drivable regions. (c) An infeasible region is projected in the moving obstacle velocity direction down lane excluding maneuvers into oncoming vehicle. In this case lane constraints have been rendered as [White] High cost instead of [Red] Infeasible due to a recovery mode triggered by the lack of progress through the intersection). (d) Within an intersection an infeasible region is created between a moving obstacle and the intersection exit matching the velocity direction.

the intersection exits. If a good exit candidate was found, a second region was projected from the obstacle toward the exit waypoint as a prediction of the traffic vehicle's intended route (Shown in Figure 12(d)). The **Motion Planner** identified, then optimized, a kino-dynamically feasible vehicle trajectory, that would move the robot toward the goal point. The module was based on the Rapidly-exploring Random Tree (RRT) algorithm (Frazzoli et al., 2002), where the tree of trajectories is grown by sampling numerous configurations randomly. A sampling-based approach was chosen due to its suitability for planning in many different driving scenarios. Uncertainty in local situational awareness was handled through rapid replanning. By design, the motion planner contains a measure of safety as the leaves on the tree of potential trajectories are always stopping locations (Figure 12(a)). Shorter trees permit lower top speeds as the vehicle must come to a stop by the end of the trajectory. In this way, if for some reason the selected trajectory from the tree became infeasible, another branch of the tree could be selected to achieve a controlled stop. The tree of trajectories was grown towards the goal by adding branches that connect to the randomly sampled points, which were then checked for feasibility and performance. This module then sends the current best vehicle trajectory, specified as an ordered list of waypoints (position, velocity, headings), to the low-level motion Controller at a rate of 10 Hz.

The **Controller** was a pure pursuit steering controller paired with a PID speed controller. It executed the low-level control necessary to track the desired path and velocity profile from the Motion Planner.

4 Team Cornell's 'Skynet'

Team Cornell's 'Skynet,' shown in Figure 13, is an autonomous 2007 Chevrolet Tahoe. *Skynet* was built and developed at Cornell University, primarily by team members returning with experience from the 2005 DARPA Grand Challenge. The team consisted of 12 core members supported by 9 part-time contributors, with experience levels including professors, doctoral and master's candidates, undergraduates, and Cornell alumni. The team was selected from a grant proposal as one of 11 research-oriented teams to receive funding from DARPA to compete in the Urban Challenge. Additional support was gained through corporate sponsors, including Singapore Technologies Kinetics, Moog, Septentrio, Trimble, Ibeo, Sick, MobilEye, The Mathworks, Delphi, and Alpha Wire.

The high-level system architecture for Team Cornell's *Skynet* is shown in Figure 14 in the form of key system blocks and data flow. These blocks form the multi-layer perception and planning / control solution chosen by Team Cornell to successfully drive in an urban environment. General descriptions of each of these blocks are given below. Detailed descriptions of the obstacle detection and tracking algorithm and the intelligent planning algorithm, both root causes of *Skynet's* behavior during the Cornell / MIT collision, are given in sections 4.2 and 4.3.

4.1 General System Architecture

Skynet observes the world with two groups of sensors. *Skynet's* position, velocity, and attitude are sensed with raw measurements collected from Global Positioning System (GPS) receivers, an Inertial Measurement Unit (IMU), and wheel encoders. These raw measurements are fused in the

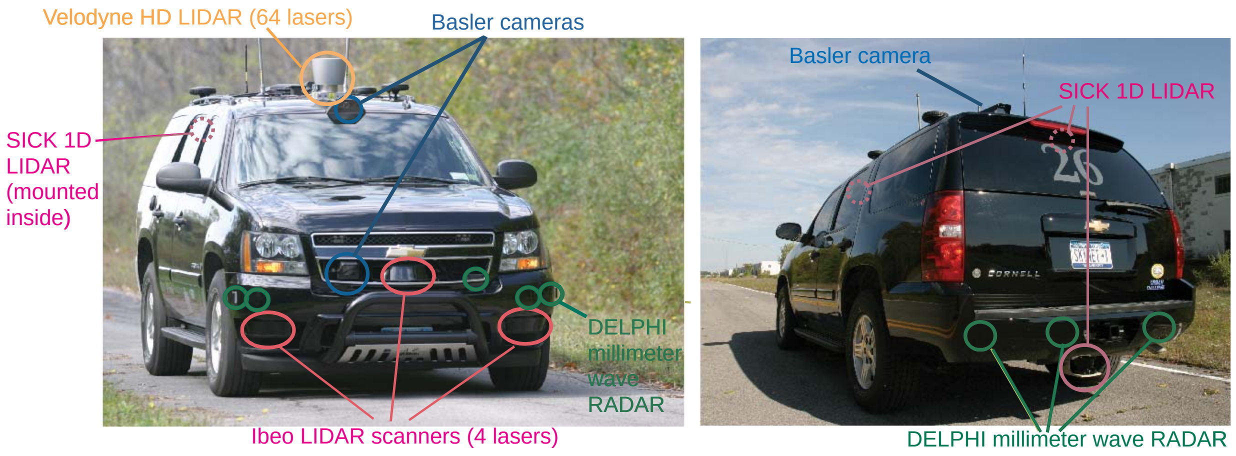

Velodyne HD LIDAR (64 lasers)
Basler cameras
SICK 1D LIDAR (mounted inside)
DELPHI millimeter wave RADAR
Ibeo LIDAR scanners (4 lasers)
Basler camera
SICK 1D LIDAR
DELPHI millimeter wave RADAR

Figure 13: Team Cornell's 'Skynet.'

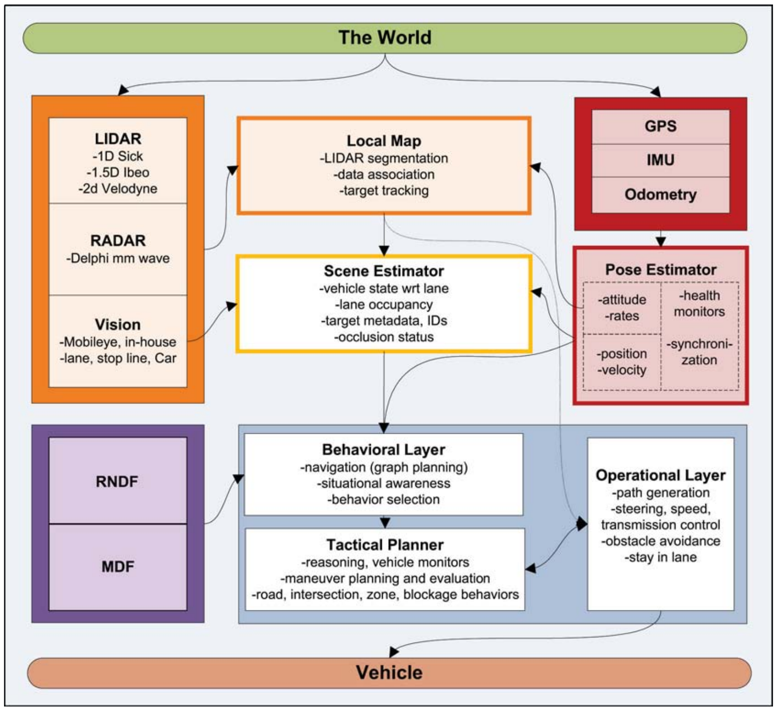


Figure 14: System architecture of Team Cornell's *Skynet*.

pose estimator, an Extended Square Root Information Filter, to produce robust pose estimates in an Earth-fixed coordinate frame. *Skynet's* external environment, defined in the Urban Challenge as parked and moving cars, small and large static obstacles, and attributes of the road itself, is sensed using a combination of laser rangefinders, radar, and optical cameras.

Skynet uses two levels of probabilistic data fusion to understand its external environment. The **Local Map** fuses laser, radar, and optical data with *Skynet's* motion estimates to initialize, locate, and track static and dynamic obstacles over time. The **Scene Estimator** then uses the local map's tracking estimates, pose estimates, and road cues from processed optical measurements to develop key statistics about *Skynet* and nearby obstacles. Two sets of statistics are generated: those concerning *Skynet*, including location with respect to the road and lane occupancy, and those concerning other obstacles, including position / velocity, an identification number, lane occupancy, car likeness, and whether each obstacle is currently occluded or not.

Planning over DARPA's Route Network Definition File (RNDF) and Mission Definition File (MDF) occurred in three layers. The topmost **Behavioral Layer** combined the RNDF and MDF with obstacle and position information from the Scene Estimator to reason about the environment and plan routes to achieve mission progress. The Behavioral Layer then selected which of four behaviors would best achieve the goal: road, intersection, zone, or blockage. The selected behavior was executed in the **Tactical Layer**, where maneuver-based reasoning and planning occur. The **Operational Layer**, the lowest level of planning, produced a target path by adjusting an initial coarse

path to respect speed, lane, obstacle, and physical vehicle constraints. *Skynet* drives the target path by converting it to a series of desired speeds and curvatures, which are tracked by feedback linearization controllers wrapped around *Skynet's* steering wheel, brake, transmission, and throttle actuators.

4.2 Obstacle Detection and Tracking

Team Cornell's obstacle detection and tracking system, called the **Local Map**, fuses the output of all obstacle detection sensors into one vehicle-centric map of *Skynet's* environment. The Local Map fuses information from three sensing modalities: laser rangefinders, radars, and optical cameras, with mounting positions shown in Figure 13. Table 1 summarizes *Skynet's* obstacle detection sensors, and Figure 15 gives a top-down view of *Skynet's* sensor coverage. All sensor measurements are fused in the local map at the object level, with each sensor measurement treated as a measurement of a single object. *Skynet's* Delphi radars and MobilEye SeeQ software (run on *Skynet's* rear-facing Unibrain optical camera) fit easily into this framework, as their proprietary algorithms transmit lists of tracked obstacles. Data from the laser rangefinders is clustered to fit into this object level framework.

The Local Map formulates obstacle detection and tracking as the task of simultaneously tracking multiple obstacles and determining which sensor measurements correspond to those obstacles (Miller and Campbell, 2007), (Miller et al., 2008). The problem is cast in the Bayesian framework of estimating a joint probability density:

$$p\left(N\left(1:k\right),X\left(1:k\right)|Z\left(1:k\right)\right) \tag{1}$$

Table 1: *Skynet's* obstacle detection sensors

Sensor	Location	Type	Rate	FoV	Resolution
Ibeo ALASCA XT	front bumper left	laser	12.5 Hz	150°	1°
	front bumper center	laser	12.5 Hz	150°	1°
	front bumper right	laser	12.5 Hz	150°	1°
Sick LMS 291	left back door	laser	75 Hz	90°	0.5°
	right back door	laser	75 Hz	90°	0.5°
Sick LMS 220	back bumper center	laser	37.5 Hz	180°	1°
Velodyne HDL-64E	roof center	laser	15 Hz	360°	0.7°
Delphi FLR	front bumper left (2x)	radar	10 Hz	15°	20 tracks
	front bumper center	radar	10 Hz	15°	20 tracks
	front bumper right (2x)	radar	10 Hz	15°	20 tracks
	back bumper left	radar	10 Hz	15°	20 tracks
	back bumper center	radar	10 Hz	15°	20 tracks
	back bumper right	radar	10 Hz	15°	20 tracks
Unibrain Fire-i 520b	back roof center	optical	15 Hz	20° – 30°	N / A

where $N(1:k)$ are a set of discrete variables assigning sensor measurements to tracked obstacles at time indices 1 through k, $X(1:k)$ are the continuous states of all obstacles being tracked at time indices 1 through k, and $Z(1:k)$ are the full set of sensor measurements at time indices 1 through k. The number of obstacles being tracked is also implicitly represented in the cardinality of the measurement assignments and obstacle states, and must be estimated by the local map. To do so, equation 1 is factorized to yield two manageable components:

$$p(N(1:k)|Z(1:k)) \cdot p(X(1:k)|N(1:k), Z(1:k)) \tag{2}$$

where, intuitively, $p(N(1:k)|Z(1:k))$ describes the task of determining the number of obstacles and assigning measurements to those obstacles, and $p(X(1:k)|N(1:k), Z(1:k))$ describes the task of tracking a known set of obstacles with known measurement correspondences. In the local map, these two densities are estimated separately using a particle filter to make Monte Carlo measurement assignments and banks of extended Kalman Filters (EKFs) to track obstacles given those assignments (Miller and Campbell, 2007), (Miller et al., 2008). The obstacles are then broadcast at 10 Hz on *Skynet's* data network. A second layer, called the **Track Generator**, combines these obstacles with *Skynet's* position estimates to generate high level obstacle metadata for the planner, including a stable identification number, whether each obstacle is stopped or shaped like a car, and whether each obstacle occupies any nearby lanes.

4.3 Intelligent Planning

Team Cornell's intelligent planning system uses *Skynet's* probabilistic interpretation of the environment to plan mission paths within the context of the rule-based road network. The planner's top level behavioral layer combined offline mission information with sensed vehicle and environment information to choose a high level behavioral state given *Skynet's* current situation. The middle level tactical layer then chose contextually-appropriate maneuvers based on the selected behavior

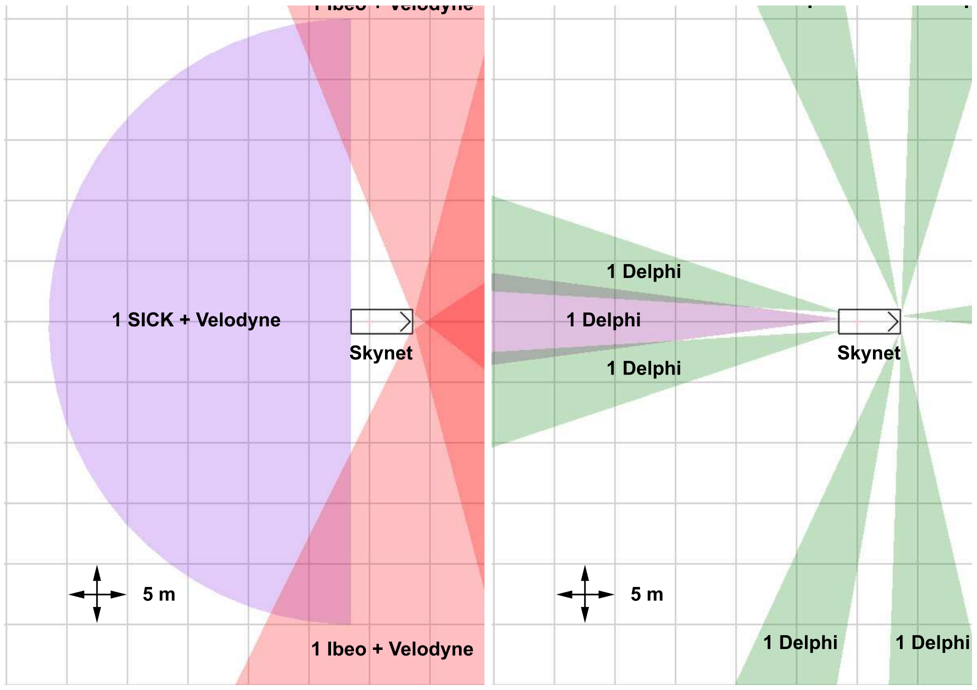


Figure 15: (left) Laser rangefinder azimuthal coverage diagram for Team Cornell's *Skynet*. (right) Radar azimuthal coverage diagram. *Skynet* faces right in both coverage diagrams. A rear-facing optical camera is not shown, nor are two laser rangefinders with vertical scan planes that detect obstacles immediately to the left and right of *Skynet*.

and the states of other nearby agents. The low-level operational layer translated these abstract maneuvers into actuator commands, taking into account road constraints and nearby obstacles. The following sections describe each of the three primary layers of the planner.

4.3.1 Behavioral Layer

The Behavioral Layer is the most abstract layer in Team Cornell's planner. Its job is to plan the fastest route to the next mission checkpoint, and then to select one of four high-level behavior states to achieve the planned route. The first part of that task, route planning, is solved using a modified version of the A* graph search algorithm (Russell and Norvig, 2003), (Ferguson et al., 2004). First, the DARPA road network was converted from the RNDF format to a graphical hierarchy of segments (Willemsen et al., 2003). The Behavioral Layer planned routes on this graphical hierarchy using dynamically calculated traversal times as costs for road partitions, lane changes, turns, and other maneuvers. After planning a route, the Behavioral Layer selected a high-level behavior state to make progress along the desired path. Four behavioral states were defined for the Urban Challenge: road, intersection, zone, and blockage, each deliberately defined as broadly as possible to promote planner stability. Each of these high -level behaviors executed a corresponding tactical component that drove *Skynet's* actions until the next behavior change.

4.3.2 Tactical Layer

When *Skynet* transitions to a new behavior state, a corresponding tactical component is executed. All components divided the area surrounding *Skynet* into regions and created monitors to detect events that might influence *Skynet's* actions. All components also accessed a common list of intelligent agents, whose behavior was monitored in the planner using estimates from the track generator. Differences between tactical components lie in the types of region monitors they use and in the actions they take in response to nearby events.

The first tactical component is the **Road Tactical**, which controls *Skynet* when it drives down an unblocked road. This component is responsible for maintaining a desired lane, evaluating possible passing maneuvers, and monitoring nearby agents. At each planning cycle, the road tactical checked agents in front of *Skynet* for speed adjustment, adjacent to *Skynet* for lane changes, and behind *Skynet* for impending collisions and reverse maneuvers (Sukthankar, 1997). Using these checks, the road tactical selected a desired speed and lane to keep. These were passed to the operational layer as a reference path.

The second tactical component is the **Intersection Tactical**, which controls *Skynet* in intersections. This component was responsible for achieving proper intersection queuing behavior and safe merging. It accomplished these goals by monitoring agent arrival times and speeds at each intersection entry, maintaining a queue of agents with precedence over *Skynet*. When the intersection monitors determine that *Skynet* is allowed to proceed, a target speed, goal point, and a polygon defining the intersection are passed along to the operational layer as a reference path.

The third tactical component is the **Zone Tactical**, which controls *Skynet* after it enters a zone. This component was responsible for basic navigation in unconstrained zones, including obstacle avoidance and alignment for parking maneuvers. The zone tactical planned over a human-annotated graph drawn on the zone during RNDF preprocessing. The graph imposed wide artificial lanes and directions of travel onto portions of the zone, allowing Skynet to treat zones as if they were roads. The zone tactical generated the same type of local lane geometry information as the road tactical to send to the operational layer as a reference path.

The final tactical component is the **Blockage Tactical**, which controls *Skynet* when obstacles block forward progress on the current route. This component is responsible for detecting and recovering from road blocks to ensure continued mission progress. Team Cornell's blockage detection and recovery relied on the Operational Layer's constrained nonlinear optimization strategy, described in section 4.3.3, to detect the location of the blockage and any possible paths through it. After initial blockage detection, the blockage tactical component proceeded through an escalation scheme to attempt recovery. First, the blockage was confirmed over multiple planning cycles, to ensure that it was not a short-lived tracking error. Second, a reverse or reroute maneuver was executed to find an alternate route on the RNDF, if one was available. If no alternate route existed, *Skynet* reset the local map and scene estimator to remove long-lived mistakes in obstacle detection. If this step fails, planning constraints are relaxed: first the admissible lane boundaries are widened, then obstacles are progressively ignored in order of increasing size. *Skynet's* recovery process escalates over several minutes in a gradual attempt to return to normal driving.

4.3.3 Operational Layer

The Operational Layer converts the Tactical Layer's reference path and speed into steering, transmission, throttle, and brake commands to drive *Skynet* along the desired path while avoiding obstacles. To accomplish this task, the Operational Layer first processes each obstacle into a planar convex hull. The obstacles are then intersected with lane boundaries to form a vehicle-fixed occupancy grid (Martin and Moravec, 1996). The A* search algorithm is used to plan an initial path through the free portion of the occupancy grid (Russell and Norvig, 2003). This initial path is then used to seed a nonlinear trajectory optimization algorithm for path smoothing.

Skynet's nonlinear trajectory optimization algorithm attempts to smooth the initial path to one that is physically drivable, subject to actuator constraints and obstacle avoidance. The algorithm discretizes the initial path into a set of n equally-spaced base points p_i, $i \in \{1,n\}$. A set of n unit-length 'search vectors' u_i, $i \in \{1,n\}$ perpendicular to the base path are also created, one for each base point. The trajectory optimizer then attempts to find a set of achievable smoothed path points $z_i = p_i + w_i \cdot u_i$, $i \in \{1,n\}$ by adjusting search weights w_i, $i \in \{1,n\}$. Target velocities v_i, $i \in \{1,n\}$ are also considered for each point, as well as a set of variables q_i^l and q_i^r, $i \in \{1,n\}$ indicating the distance by which each smoothed path point z_i violates desired spacings on the left and right of *Skynet* created from the list of polygonal obstacles. Search weights, velocities, and final obstacle spacings are chosen to minimize the cost function J:

$$\begin{aligned} J\left(w_i, v_i, q_i^l, q_i^r\right) &= \alpha_c \sum_{i=2}^{n-1} c_i^2 + \alpha_d \sum_{i=2}^{n-2} \left(c_{i+1} - c_i\right)^2 + \alpha_w \sum_{i=1}^{n} \left(w_i - w_i^t\right)^2 \\ &+ \alpha_q \sum_{i=1}^{n} \left(q_i^l + q_i^r\right) + \alpha_a \sum_{i=1}^{n-1} a_i^2 - \alpha_v \sum_{i=1}^{n} v_i \end{aligned} \tag{3}$$

where α_c, α_d, α_w, α_q, α_a, and α_v are tuning weights, c_i is the approximated curvature at the i^{th} path point, w_i^t is the target search weight at the i^{th} path point, and a_i is the approximated forward vehicle acceleration at the i^{th} path point. This cost function is optimized subject to a set of 6 rigid path constraints:

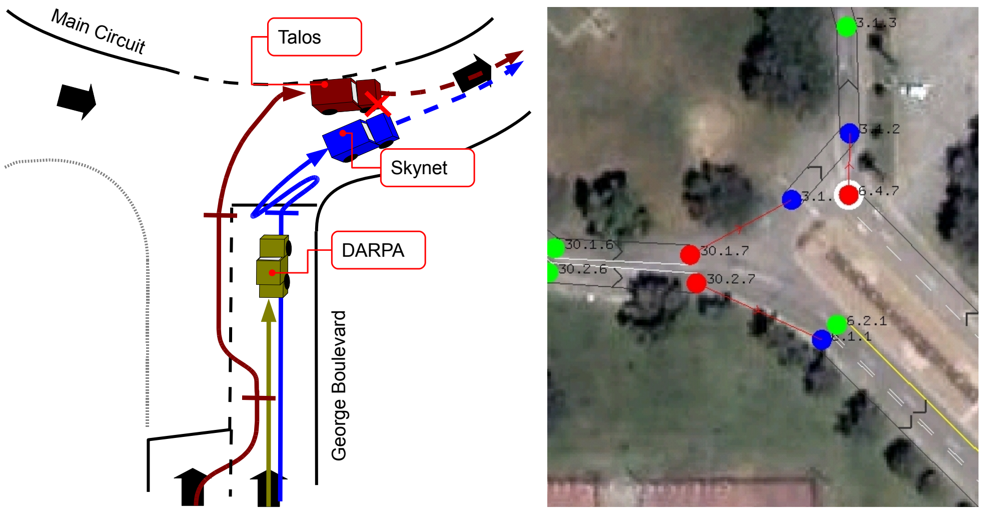
1. Each search weight w_i cannot push the smoothed path outside the boundary polygon supplied by the tactical layer.
2. Each obstacle spacing variable q_i^l and q_i^r cannot exceed any obstacle's minimum spacing requirement.
3. Curvature at each path point cannot exceed *Skynet's* maximum turning curvature.
4. Total forward and lateral vehicle acceleration at each path point cannot exceed assigned limits.
5. Each search weight w_i and set of slack variables q_i^l and q_i^r must never bring *Skynet* closer to any obstacle than its minimum allowed spacing.
6. The difference between consecutive path weights w_i and w_{i+1} must not exceed a minimum and maximum.

Additional constraints on initial and final path heading are also occasionally included to restrict the smoothed path to a particular end orientation, such as remaining parallel to a lane or a parking spot.

The constrained optimization problem is solved using LOQO, an off-the-shelf nonlinear non-convex optimization library. Two optimization passes are made through each base path to reach a final smoothed path. The first step of the smoothed path is then handed to two independent low-level tracking controllers, one for desired speed and one for desired curvature. The optimization is restarted from scratch at each planning cycle, and is run at 10 Hz.

5 The Collision

Undoubtedly the most observed incident between robots during the Urban Challenge was the low-speed collision of *Talos* with *Skynet*. The location of the incident and a diagram of the accident progression are shown in Figure 16.



(a) (b)

Figure 16: (a) Diagram of the incident. (b) The collision took place while the vehicles traversed the intersection from waypoint (6.4.7) to (3.1.2).

The collision between *Skynet* and *Talos* occurred during the second mission for both teams. Both vehicles had driven down Washington Boulevard and were attempting to merge on to Main Circuit to complete their latest sub-mission. *Skynet* drove down George Boulevard and was the first to arrive at the intersection. The vehicle paused, moved forward on to Main Circuit (around two car lengths), and then came to a stop. It backed up about three car lengths, stopped, drove forward a car length, stopped again before finally moving forward just as *Talos* was approaching Main Circuit. *Talos* was behind *Skynet* and *Skynet's* chase vehicle on approach to the intersection. *Talos* then passed the queuing *Skynet* chase vehicle on the left. *Talos* then stopped beside the chase vehicle while *Skynet* was backing up back over the stop line. When *Skynet* moved forward again, *Talos* drove up and came to a stop at the stop line of the intersection. *Talos* then drove out to the left of *Skynet* as if to pass. *Talos* was along side *Skynet* in what was looking to be a successful passing maneuver, when *Talos* turned right, pulling close in front of *Skynet*, which was now moving forward.

Next, in Sections 6 and 7, we will branch off and look at the collision from inside the *Skynet* and *Talos* software.

6 The Collision from inside *Skynet*

UCE spectators characterized *Skynet* as having three erratic maneuvers in the seconds leading up to its collision with *Talos*. First, *Skynet* stuttered through its turn into the south entrance of the traffic circle, coming to several abrupt stops. Second, *Skynet* drove backward after its second stop, returning almost fully to the stop line from which it had just departed. Finally, *Skynet* stuttered through the turn once again, ignoring *Talos* as it approached from behind, around to *Skynet's* driver side, and finally into a collision near *Skynet's* front left headlight. Sections 6.1, 6.2, and 6.3 describe, from a systems-level perspective, the sequence of events causing each erratic maneuver.

6.1 Stuttering Through the Turn

Although it did not directly cause the collision, *Skynet's* stuttering through its turn into the traffic circle was one of the first erratic behaviors to contribute in the collision. At its core, *Skynet's* stuttering was caused by a complex interaction between the geometry of the UCE course and its GPS waypoints near the turn, the probabilistic obstacle detection system discussed in section 4.2, and the constraint-based planner discussed in section 4.3.3. First, Team Cornell defined initial lane boundaries by growing polygonal admissible driving regions from the GPS waypoints defining the UCE course. This piecewise-linear interpretation of the lane works best when the lane is straight or has shallow curves: sharp turns can yield polygons that exclude significant portions of the lane. The turn at the southern entrance to the traffic circle suffered from this problem acutely, as the turn is closely bounded on the right by concrete barriers and a spectator area. Figure 17 shows that these concrete barriers occupied a large region of the lane polygon implied by the DARPA waypoints. The resulting crowded lane polygon made the turn difficult: *Skynet's* constraint-based Operational Layer, described in section 4.3.3, would

not generate paths that drive outside the lane polygon. With space already constrained by *Skynet's* internal lane polygon, small errors in absolute position or obstacle estimates might make the path appear infeasible.

Path infeasibility caused by these types of errors resulted in *Skynet's* stuttering through the south entrance to the traffic circle. At the time leading up to the collision, small variations in clusters of laser rangefinder returns and Monte Carlo measurement assignments in the local map caused *Skynet's* path constraints to change slightly from one planning cycle to the next. In several cases, such as the one shown in Figure 18, the constraints changed to make *Skynet's* current path infeasible. At that point *Skynet* hit the brakes, as the Operational Layer was unable to find a feasible path along which it could make forward progress.

In most cases, variations in the shapes of obstacle clusters and Monte Carlo measurement assignments, like the one shown in Figure 18, clear in one or two planning cycles: for these, *Skynet* taps

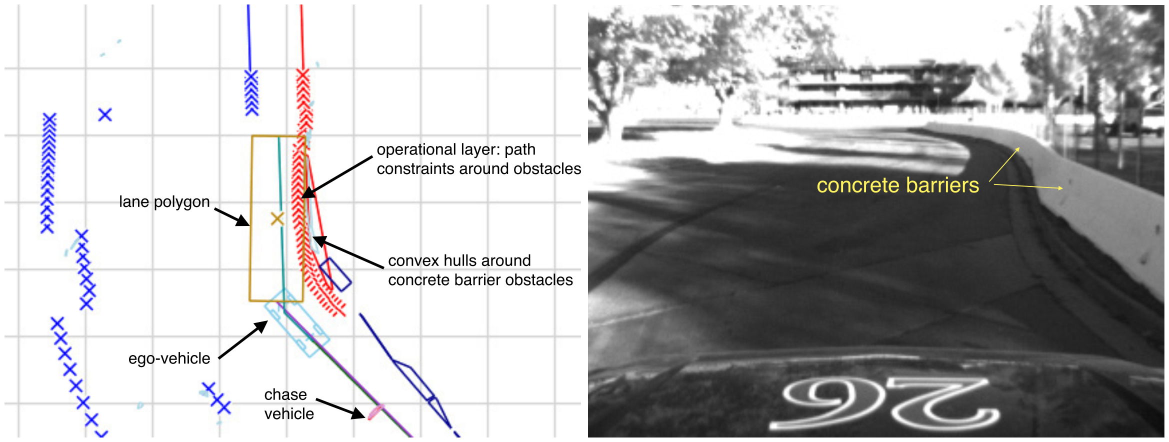

operational layer: path constraints around obstacles
lane polygon
convex hulls around concrete barrier obstacles
ego-vehicle
chase vehicle
concrete barriers
26

Figure 17: (left) The lane polygon implied by piecewise-linear interpolation of DARPA waypoints in the turn near the south entrance to the traffic circle. Obstacle constraints from nearby concrete barriers occupy a significant portion of the lane polygon. (right) *Skynet* camera view of the concrete barriers generating the constraints.

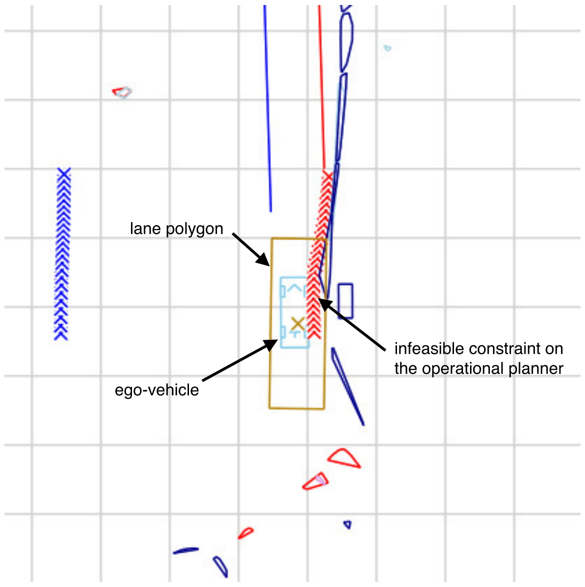


Figure 18: Small variations in *Skynet's* perception of a concrete barrier cause its planned path to become infeasible.

the brakes before recovering to its normal driving mode. These brake taps were generally isolated, but were more deleterious near the traffic circle for two reasons. First, the implied lane polygons forced *Skynet* to drive close to the concrete barriers, making it more likely for small mistakes to result in path infeasibility. Second, *Skynet's* close proximity to the concrete barriers actually made clustering and local map mistakes more likely: Ibeo laser rangefinders and Delphi radars tended to produce more false detections when within 1.5 m of an obstacle. The interaction of these factors produced the stuttering behavior, which happened several times at that corner during the UCE.

6.2 Reversing Toward the Stop Line

Occasionally, variations in obstacle clusters and poor Monte Carlo measurement assignments in the local map are more persistent: in these cases phantom obstacles may appear in the lane, blocking forward progress for several seconds. In these failures the local map typically does not have enough supporting sensor evidence to delete the phantom obstacle immediately, and allows it to persist until that evidence is accumulated. When this happens, *Skynet* considers the path blocked, executing the blockage recovery tactical component to deal with the situation. Blockage recovery was activated 10 times over the 6 hours of the UCE.

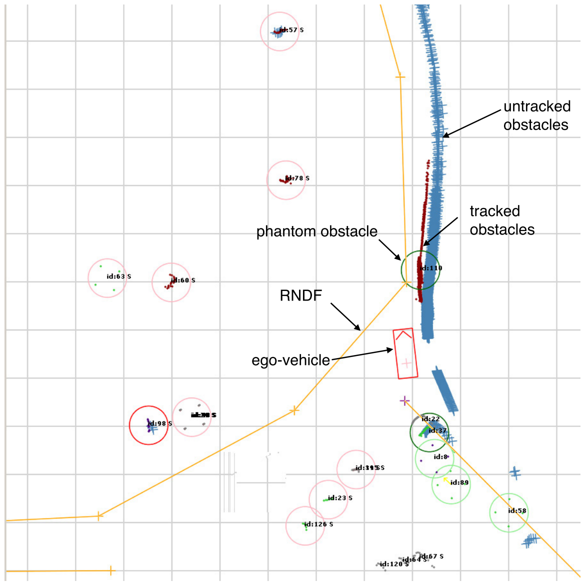


Figure 19: A measurement assignment mistake causes a phantom obstacle to appear, momentarily blocking *Skynet's* path.

One of the 10 blockage recovery executions occurred immediately prior to *Skynet's* collision with *Talos*. In this scenario, a measurement assignment mistake caused a phantom obstacle to appear part-way into *Skynet's* lane. The phantom obstacle, shown in Figure 19, caused *Skynet* to execute an emergency braking maneuver. The phantom obstacle was deleted after approximately 2 seconds, but the adjustments to the Operational Layer's constraints persisted long enough for the Operational Layer to declare the path infeasible and the lane blocked. The mistake sent *Skynet* into blockage recovery. In blockage recovery, the Operational Layer recommended the Tactical Layer reverse to reposition itself for the turn. The Tactical Layer accepted the recommendation, and *Skynet* reversed one vehicle length to reposition itself.

6.3 Ignoring *Talos*

After the reverse maneuver described in section 6.2, *Skynet* still had not completed the turn necessary to continue with its mission. The planner therefore remained in its blockage recovery state, though recommendation and completion of the reverse maneuver left it in an escalated state of blockage recovery. In this state the Tactical Layer and Operational Layer once again evaluated the turn into the traffic circle, this time ignoring small obstacles according to the blockage recovery protocol described in section 4.3.2. The Operational Layer decided the turn was feasible, and resumed forward progress. Although the Local Map produced no more phantom obstacles for the duration of the turn, small errors in laser rangefinder returns once again forced the Operational Layer to conclude that the path was infeasible. At this point, the Tactical Layer escalated to its highest state of blockage recovery, removing constraints associated with lane boundaries. Figure 20 shows this escalation from *Skynet's* normal turn behavior to its decision to ignore lane boundaries.

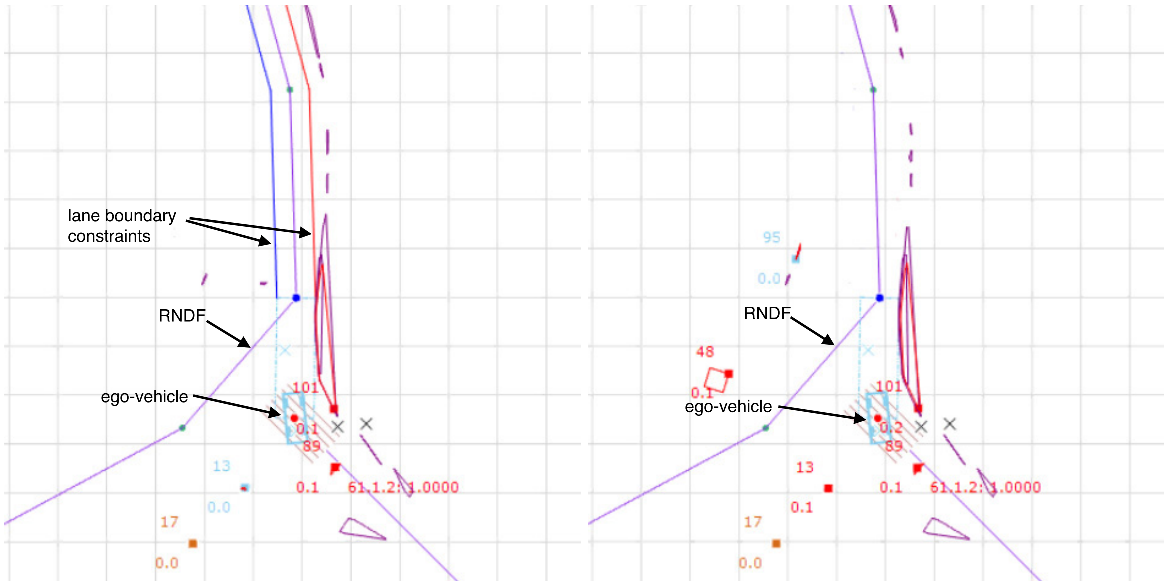


Figure 20: (left) *Skynet* resumes its turn after a reverse maneuver. (right) Perceiving the turn infeasible a second time, *Skynet* drops constraints associated with lane boundaries.

Unfortunately, *Skynet* still perceived its goal state as unreachable due to the nearby concrete barriers. At the highest level of blockage recovery, however, *Skynet* was deliberately forbidden to execute a second reverse maneuver to prevent an infinite planer loop. Instead, it started a timer to wait for the error to correct itself, or barring forward progress for several minutes, to reset the local map and eventually the planner itself. Neither of these soft resets would be realized, however, as *Talos* was already weaving its way behind as *Skynet* started its timer.

While *Talos* passed behind and then to the left of *Skynet*, the Operational Layer continued to believe the forward path infeasible. Coincidentally, the path was perceived as feasible just as *Talos* pulled out to pass *Skynet* on the left. With the path momentarily feasible, *Skynet* began to drive forward as *Talos* passed on its left. Here *Skynet's* Tactical Layer ignored *Talos*, because *Talos* drove outside the piecewise-linear polygonal lane boundary, as shown in Figure 21. *Skynet's*

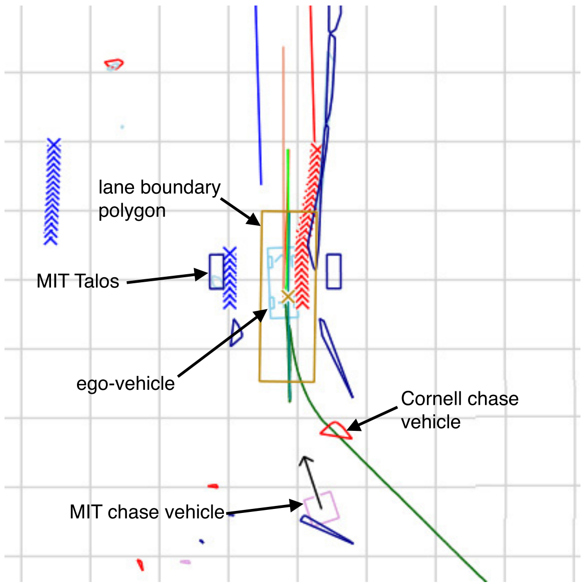


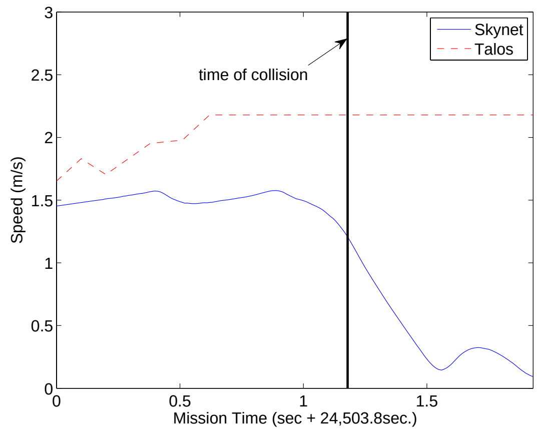
Figure 21: Skynet ignores *Talos* as it drives outside *Skynet's* polygonal lane boundary.

Operational Layer also ignored *Talos*, as *Talos* did not constrain the target path in front of *Skynet* in any way. Once *Talos* passed to *Skynet's* left, *Talos* was no longer detected as a moving obstacle; *Skynet's* sideways-facing Sick LMS-291s are mounted with a vertical scan plane and provide only weak position information and no velocity information. The Local Map began tracking *Talos* as a moving obstacle only 1 second before the collision, when it entered into view of *Skynet's* forward-mounted Ibeo ALASCA XTs. Unfortunately, with concrete barriers on *Skynet's* right and *Talos* approaching on its left, no evasive maneuver was available. At that point the collision was inevitable, and the collision occurred.

Figure 22 shows the speed and heading, as estimated on *Skynet*, for both the *Skynet* and *Talos* vehicles. Approximately 0.5 sec before collision, the speed and heading estimates for *Talos* remain constant, which is the time that they are stopped being tracked. *Skynet* did not change its heading or velocity before the collision, indicating that no adjustments were made to the *Talos* movements. Finally, after impact, there is a fast change in *Skynet's* heading, indicating the collision, and its velocity decreases quickly to zero soon after.

7 The Collision from inside *Talos*

The incident from the *Talos'* viewpoint is shown in Figures 23, 24 and 25. Figure 23 shows that earlier along George Boulevard, the road was dual-lane. *Talos* was going faster along the straight road than the *Skynet* chase vehicle, so *Talos* passed to the left of the chase vehicle (Figure 23(a)). At the end of Washington Boulevard, the road merged (via cones on the left) into a single lane on the right. *Talos* did not have room to merge right in front of the chase vehicle, so *Talos* slowed to a stop while the *Skynet* chase vehicle moved ahead. When space was available, *Talos* merged behind the *Skynet* chase vehicle (Figure 23(b)). *Skynet* and the chase vehicle then come to a stop at the intersection (Figure 23(c)).



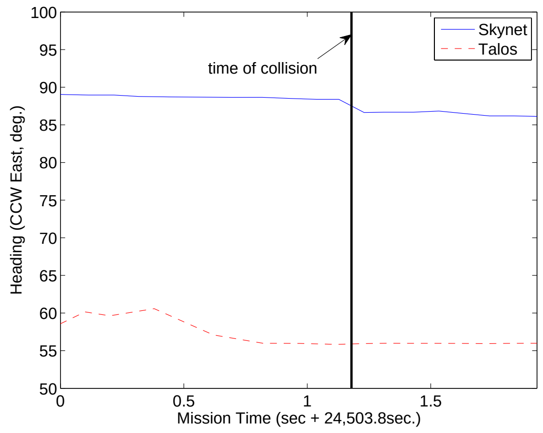


Figure 22: From *Skynet* logs: Speed (left) and heading (right) for both *Skynet* and *Talos* just before and after collision. Flat line in *Talos'* plot indicates where *Skynet* stopped tracking *Talos*.

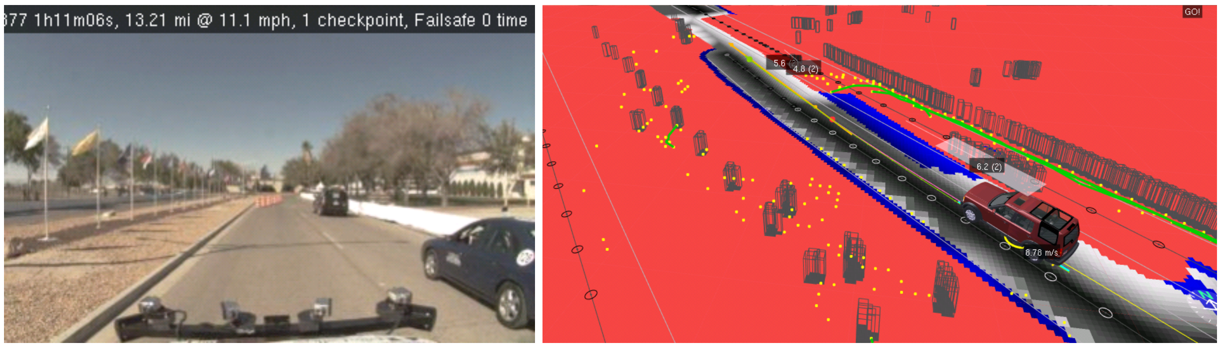
In Figure 24 we see that at first, *Talos* stops behind the chase vehicle. However, the lane width is sufficient that *Talos* soon finds a path to the left of the chase vehicle (Figure 24(a)). In this case *Talos* is not in a passing mode; it has simply found room on the left-hand side of the current lane to squeeze past the DARPA chase vehicle.

7.1 Wide lane bug

The lane is significantly wider to the left because of a Drivability Map construction bug. As described in Section 3, lanes are carved out of the lane-cost map like valleys through a plateau. Adjacent lanes carved out often caused small islands remaining between the valleys. These islands were addressed by explicitly planing down the region between adjacent lanes. This strategy worked well in general however in this case the road merges down to one lane shortly before the intersection. The adjacent lane is not rendered after the merge, which is correct. However, the planing operation was done all the way along the right lane past the merge point. The effect of the planing alone makes the road 3 meters wider on the left than it would otherwise be. Without the extra width, *Talos* would have been forced to queue behind the DARPA chase vehicle.

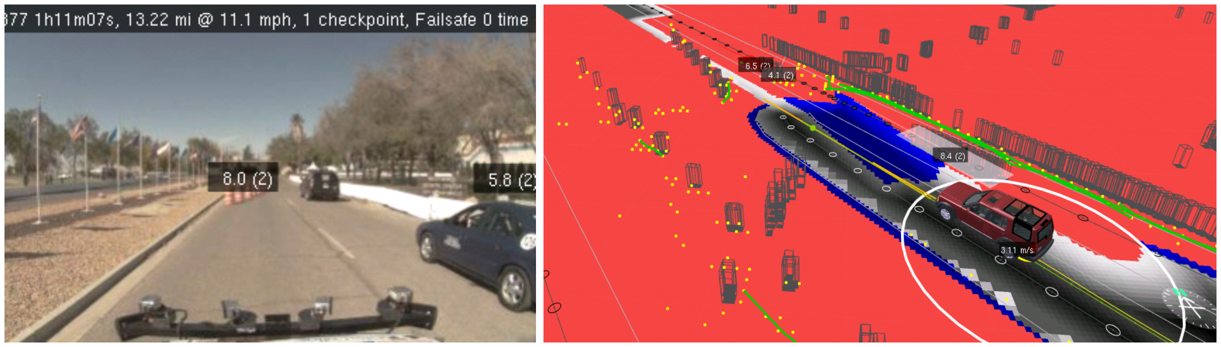
7.2 At the intersection

Figures 24(b) & (c) show how *Talos* pulled out and drove around to the left of the chase vehicle. The robot had a motion plan which was attempting to reach a goal point on the stop line of the intersection. *Talos* was beside the chase vehicle when *Skynet* backed up and occupied *Talos'* goal position. *Talos* came to a stop, unable to drive through the restricted region to get to the goal. In the visualization, *Skynet* did not have a restricted region in front and behind the vehicle. This was because *Skynet* was within the intersection. Obstacles detected inside the intersection did not have restricted regions because the heuristic was that obstacles inside intersections were things like sign posts, traffic islands and encroaching trees. *Skynet* then moved forward again, making *Talos'* goal position clear. *Talos* drove to the stop line. Although now adjacent to *Skynet's* chase



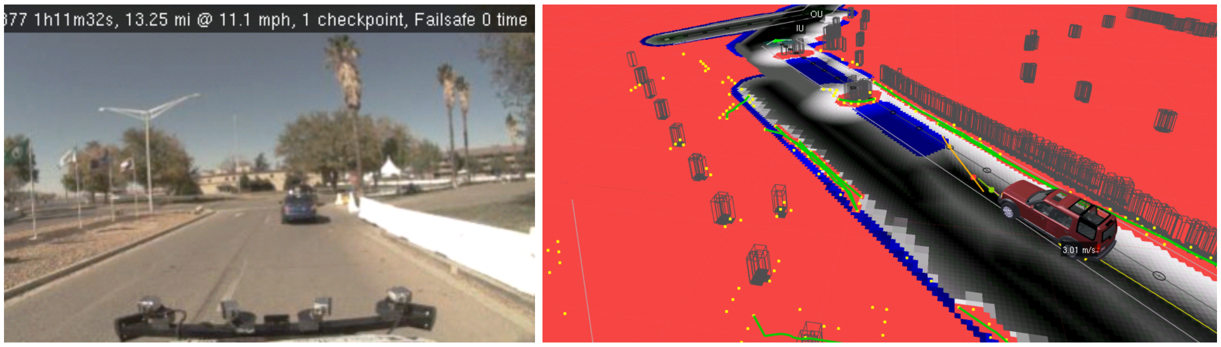
377 1h11m06s, 13.21 mi @ 11.1 mph, 1 checkpoint, Failsafe 0 time
5.6
4.8 (2)
6.2 (2)
8.78 m/s
GO!

(a)



377 1h11m07s, 13.22 mi @ 11.1 mph, 1 checkpoint, Failsafe 0 time
8.0 (2)
5.8 (2
6.5 (2)
4.1 (2)
8.4 (2)
3.11 m/s

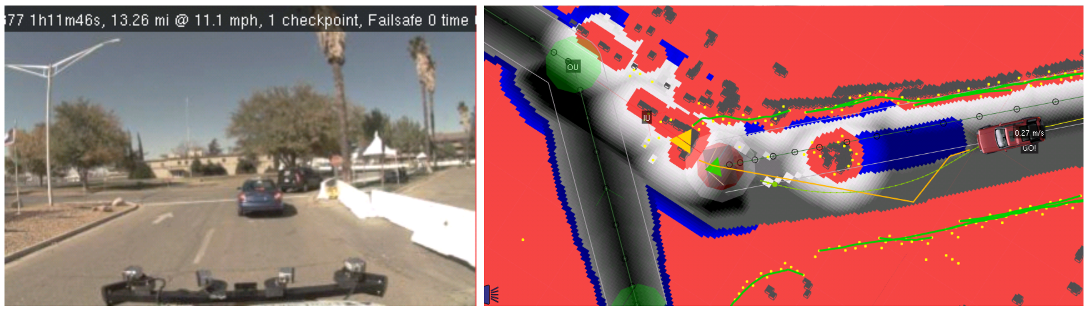
(b)



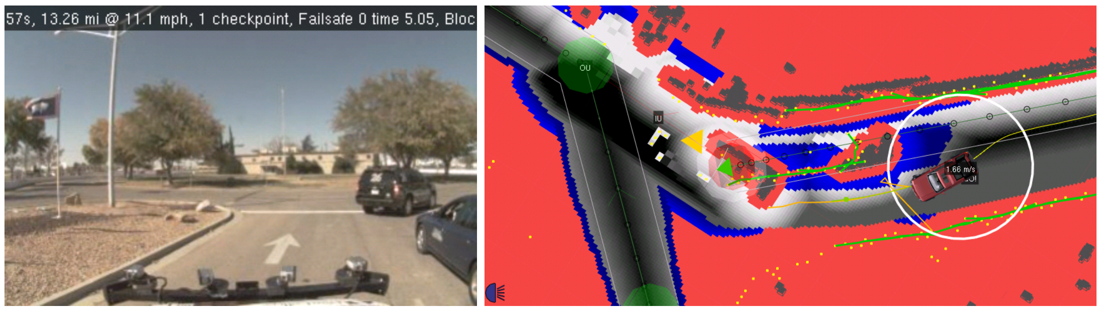
377 1h11m32s, 13.25 mi @ 11.1 mph, 1 checkpoint, Failsafe 0 time
3.01 m/s

(c)

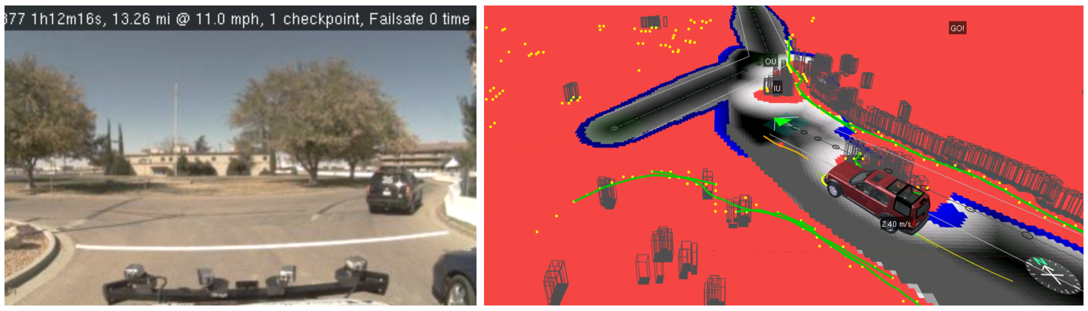
Figure 23: *Talos'* view of the lead-up to the *Skynet*-*Talos* incident. (a) *Talos* starting to pass *Skynet's* chase vehicle. (b) *Talos* is forced to slow down and merge behind the *Skynet* chase vehicle. (c) *Talos* queues behind the Skynet chase vehicle.


377 1h11m46s, 13.26 mi @ 11.1 mph, 1 checkpoint, Failsafe 0 time
OU
IU
0.27 m/s
GO!

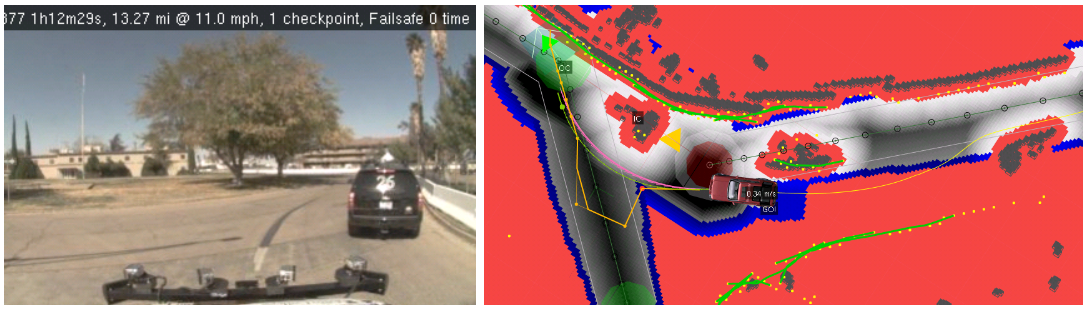
(a)


57s, 13.26 mi @ 11.1 mph, 1 checkpoint, Failsafe 0 time 5.05, Bloc
OU
IU
1.66 m/s
GO!

(b)


377 1h12m16s, 13.26 mi @ 11.0 mph, 1 checkpoint, Failsafe 0 time
GO!
OU
IU
2.40 m/s

(c)


377 1h12m29s, 13.27 mi @ 11.0 mph, 1 checkpoint, Failsafe 0 time
OC
IC
0.34 m/s
GO!

(d)

Figure 24: Lead-up to the *Skynet*-*Talos* incident. (a) *Talos* finds a route around the chase vehicle. (b) *Skynet* backs up onto *Talos'* goal position, *Talos* brakes. (c) *Skynet* advances again. *Talos* passes the chase vehicle. (d) *Talos* yields at the intersection. There are no moving vehicles nearby, so it proceeds.

vehicle, *Talos* wasn't in a failsafe mode. The artificially widened lane permitted *Talos* to drive up to the intersection as it would a passing parked car or any other object on the side of the road not blocking the path. At the intersection *Talos* followed the standard procedure of giving precedence to obstacle/vehicles approaching on the left. There was no moving or static obstacles in the intersection to the left of *Talos* on Main Circuit, so the software moved the short term goal point to the exit of the intersection (waypoint 3.1.2) and *Talos* proceeded.

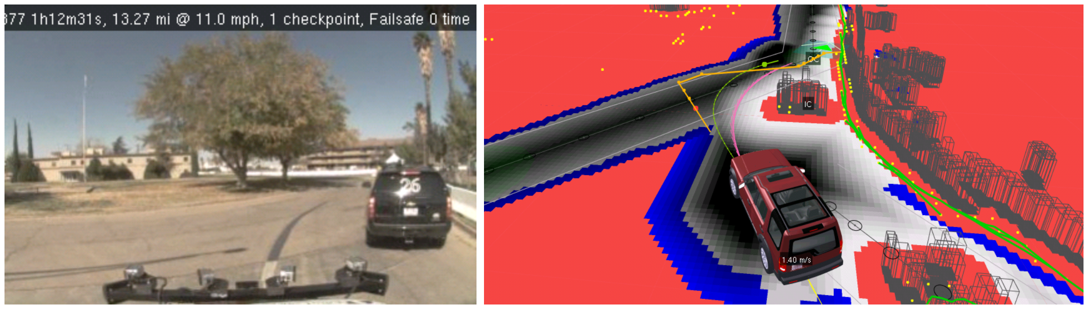
7.3 The collision

Finally, Figures 24(d) and 25(a) show *Talos* planning a path out to the left through a region that had a low cost by avoiding *Skynet* (which *Talos* perceived as a static obstacle). *Talos'* goal point moved further down *Main Circuit*, requiring the robot to find a trajectory that would have an approach angle to the lane shallow enough to drive within the lane boundaries of Main Circuit. *Talos* was now inside the intersection bounding box. *Talos* planned a path around the "*Skynet* static object" and down *Main Circuit* within the lane boundaries. The path was close to *Skynet* so the route had a high cost but was physically feasible. *Talos* drove between *Skynet* (on the right) and the lane boundary constraint (on the left). *Talos* then pulled to the right so that it could make the required approach angle to enter the lane. Had *Skynet* remained stationary at this point *Talos* would have completed the passing maneuver successfully. In Figure 25(b), we can see that *Skynet* starts moving forward. Had *Skynet* been moving faster (i.e., $>$ 3m/s instead of 1.5m/s), a moving obstacle track would have been initiated in the *Talos* software and a "no-go" region would have been extruded in front of the vehicle. This region would have caused *Talos* to yield to *Skynet* (similar to what occurred with *Odin* and *Talos* in Section 2.5). Instead *Talos* turns to the right to get the correct approach angle to drive down the lane; *Skynet* moves forward; the robots collide (Figure 25(c)).

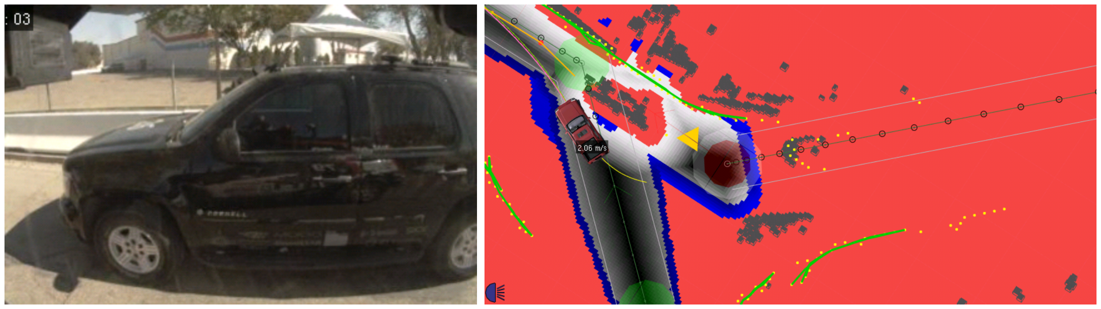
7.4 Clusters of static objects

Talos perceived *Skynet* as a cluster of static objects. The positions of the static objects evolved over time. This may sound strange, however it is not uncommon for a cluster of static obstacles to change shape as the ego-vehicle position moves. It could be due to, for instance, more of an object becoming visible to the sensors. For example, the extent of the concrete barrier detected on the right of *Talos* in Figures 23(a),(b)& (c) varied due to sensor range and aspect in relation to the ego-vehicle. Because the software treated *Skynet* as a collection of static objects instead of a moving obstacle and no forward prediction was made on *Skynet's* motion. *Talos* was driving between a lane constraint on the left and a collection of static objects on the right. If *Skynet* had not moved *Talos* would have negotiated the gap just as it had to avoid K-rails and other static objects adjacent to the lanes throughout the day. Instead, unexpectedly for *Talos*, *Skynet* moved forward into the planned path of *Talos*. Without a forward-motion prediction of *Skynet*, by the time *Skynet* was in *Talos'* path, *Talos* was unable to come to a stop before colliding.

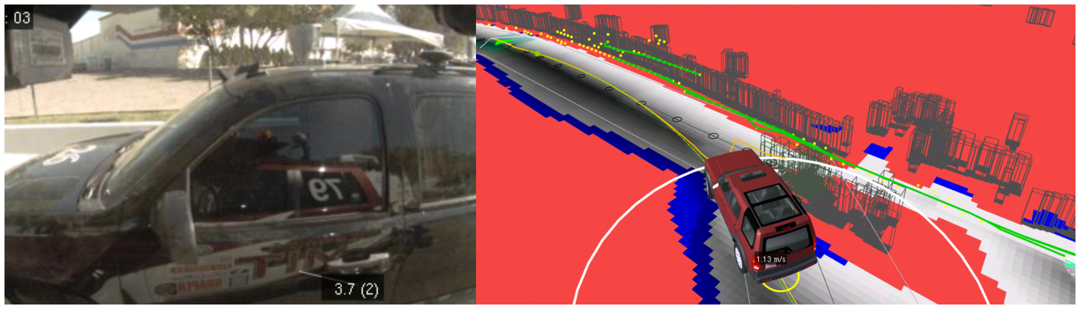
Figure 26 shows the vehicle state during the collision. Talos had straightened it's wheels to around 9° to the right, was traveling around $2m/s$. The vehicle detected that the motion planning tree had been severed $550msec$ before the collision, it was replanning, no evasive maneuver was performed yet. The vehicle was coasting $150msec$ later the DARPA Paused the vehicle. At the collision *Talos* was moving at $1.5m/s$ dropping to zero $750msec$ after the initial collision. In the log visualization


377 1h12m31s, 13.27 mi @ 11.0 mph, 1 checkpoint, Failsafe 0 time
1.40 m/s

(a)



(b)



(c)

Figure 25: *Skynet-Talos* incident. (a) *Talos* plans a route around *Skynet*, which appears as a static object. (b) While *Talos* is passing, *Skynet* begins to move. (c) While applying emergency braking, *Talos* turns into *Skynet*.

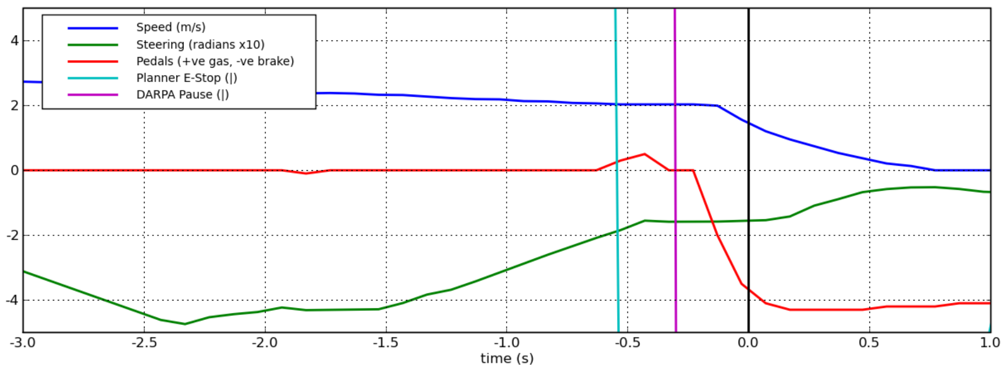


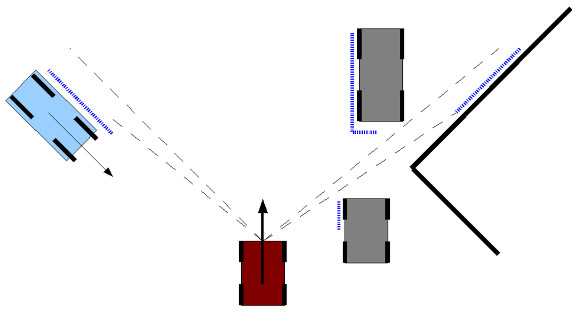
Figure 26: *Talos'* speed, wheel angle and pedal gas and brake positions during collision. Time 0.0 is the initial collision. Motion Planner E-Stop was at $-550msec$. DARPA Pause at $-400msec$. The vehicle came to rest after $750msec$.

Talos is pushed slightly forward and to the left of it's heading by the impact (about $0.3m$).

The contributing factors of *Talos'* behavior can be decomposed as: the inability to track slow-moving objects, the use of an moving-obstacle model versus explicit vehicle classification and the dominant influence of lane constraints on motion planning & emergency path diversity. The other contributing factor, the Drivability Map rendering bug which widened the lane to allow *Talos* to attempt to drive around instead of queue, is a test-coverage issue and holds little to analyze further.

7.5 Inability to track slow-moving objects

In the obstacle detection system all tracked objects had a velocity component. Obstacles with small velocities were considered static and the velocity information was assumed to be noise. In addition to sensor noise there was also noise injected by vehicle-object pose changes, causing more of the object to be revealed. The new returns would be clustered with the existing obstacle cluster, moving the center of mass of the obstacle and making it appear to have moved. An intelligent clustering technique to handle new obstacle-hits or just new hits helps in this case, but in some cases there is no information to detect this case. The lidar aperture problem is an acute case of this issue. Figure 27 illustrates the sensor aperture problem in this case for a horizontally mounted lidar in the detection of an approaching vehicle. As the robot moves forward or the traffic vehicles move, the building on the right produces returns consistent with a rapidly approaching vehicle. Several groups have attempted to counter this problem using algorithms to determine the shadowing of distant objects by near-field returns (Thrun et al., 2006). However, with more complex sensor characteristics, such as the 64 laser Velodyne sensor, and more complex scene geometries for urban environments, these techniques become difficult to compute as a flat obstacle occlusion map is no longer sufficient, since obstacle height must be considered.



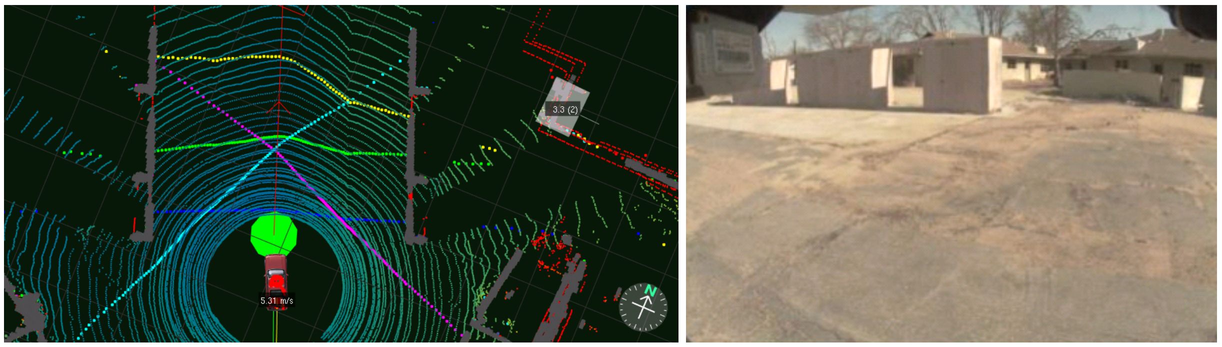


Figure 27: (a) Illustration of lidar aperture problem. The building on right generates a lidar return indistinguishable from the fast-approaching vehicle on the left. (a) Aperture problem inducing phantom moving obstacles. (b) walls entering the *White Zone* generate phantom moving vehicles from building returns.

7.6 Moving Obstacles versus Explicit Vehicle Classification

As described in Section 3, the MIT vehicle did not explicitly detect vehicles. Instead, objects in the scene were classified either as static or moving obstacles. Moving obstacles were rendered with exclusion regions in the direction of travel along the lane. The decision to use moving obstacles was taken to avoid the limitations of attempting to classify the sensing data into “vehicle” or “non-vehicle” classes. The integrated system was fragile however as classification errors or outages caused failures in down stream modules blindly relying on the classifications and their persistence over time. Up until and including the MIT site visit in June 2007, the software did attempt to classify vehicles based on size and lane proximity. During one mission lane precedence failed due to sensor noise, causing a vehicle to be lost momentarily and then reacquired. The reacquired vehicle had a different ID number, making it appear as a new arrival to the intersection, so *Talos* incorrectly went next. In reaction to this failure mode, Team MIT migrated to use the concept of static and moving obstacles instead. Sensor data classification schemes, by requiring a choice to be made, introduce the potential for false positive and false negative classifications. Much care can be placed in reducing the likelihood of mis-classifications, however the classification errors can almost always still occur. Developers often design down-stream applications to be overconfident in the classes assigned to objects. Avoiding the assignment of "vehicle"/"non-vehicle" classes to detected objects was an attempt to cut down assumptions made by the down-stream applications interpreting the detected obstacle data. The assumptions were made in relation to the strict definition of "static" and "moving" obstacles instead. On the whole this approach scaled well with the additional complexity of the final race. The apparent gap was in the correct treatment of active yet stationary vehicles. The posture of *Skynet* is not very different from the stationary cars parked in the Gauntlet of Area B during the qualifiers.

7.7 Lane Constraints and Emergency Path Diversity

In the nominal situation, the tree of trajectories end in stopped states, so that *Talos* always knows how to come to a safe stop. When *Talos* is moving and the planner cannot find any feasible safe path from the current configuration (possibly due to a change in the perceived environment caused by sensing noise or dynamic obstacles that change the constraints) the planner generates a emergency braking plan. This emergency plan consists of the steering profile of the last feasible path and a speed profile with the maximum allowable deceleration. Before the collision (Figure 25(b)), the tree of trajectories was going towards the target further down the road. When the gap between the left lane boundary and *Skynet* was narrowed as the *Skynet* moved forward, no feasible plan was found that stopped *Talos* safely.When no feasible solution is found, a better approach would be to prioritize the constraints. In an emergency situation, satisfying lane constraints is not as important as avoiding a collision. Therefore, when generating an emergency plan, the planner could soften the lane constraints (still using a relatively high cost) and focus on ensuring collision avoidance with the maximum possible braking.

8 Discussion

Neither vehicle drove in a manner "sensible" to a human driver. On a day of fine weather and good visibility *Skynet* backed up in a clear intersection and started to accelerate when another vehicle was closing in. *Talos* passed a vehicle instead of queuing in a single-lane approach, then pulled in much too close to an active vehicle in an intersection.

To summarize, contributing factors identified in the two vehicles' software were:

- *Talos'* lane-rendering bug permitting *Talos* to pass the DARPA chase vehicle;
- *Talos'* inability to track slow-moving objects, and
- *Skynet's* sensor data associations inducing phantom objects;
- *Talos'* failure to anticipate potential motion of an active stationary vehicle;
- *Skynet's* failure to accommodate the motion of an adjacent vehicle in an intersection;
- *Talos'* overly constrainted motion due to target lane constraints.
- *Skynet's* lane representation narrowing the drivable corridor;

Apart from the lane-rendering problem, these factors are more than just bugs: they reflect hard trade-offs in road environment perception and motion planning.

8.1 Sensor data clustering

Skynet's phantom obstacles and *Talos'* inability to track slow-moving objects represent the downsides of two different approaches to address the same problem of sensor data clustering. Team Cornell chose to estimate the joint probability density across obstacles using Monte Carlo measurement assignments to associate sensor data with objects (Section 4.2). The consequence was that sometimes the associations would be wrong, creating false positives. Team MIT found lidar data clustering too noisy to use for static objects. Instead, relying on its sensor-rich vehicle, the accumulator array with a high entropy presented static objects to motion planning directly. Once the velocity signal was sufficiently strong the clustered features robustly tracked moving objects. A high threshold was set before moving obstacle tracks were reported to suppress false positives. The consequence was until the threshold was passed, there was no motion prediction for slow moving objects.

8.2 Implicit and Explicit Vehicle Detection

The treatment of vehicles in the road environment must extend past the physics-based justification of obstacle avoidance due to closing rate. For example, humans prefer never to drive into the longitudinal path of an occupied vehicle, even if it is stationary. In Section 2 we mentioned how the DARPA chase vehicle driver preferred to drive on the curb than in front of the Paused *Skynet* vehicle.

Many teams in the contest performed implicit vehicle detection using the object position in the lane and size to identify vehicles(Leonard et al., 2008; Miller et al., 2008; Stanford Racing Team, 2007). Moving objects detected with lidar or using radar Doppler velocity were also often assumed to be vehicles. To prevent identified vehicles being lost, several teams had a “was moving” flag associated with stationary tracked objects, such as queuing vehicles, that had once been observed moving(Tartan Racing, 2007). It is not difficult to imagine a case where a vehicle would not have been observed moving and the vehicle size and position rules of thumb would fail. Some teams also used explicit vehicle detectors such as the Mobileye SeeQ system. However, explicit vehicle detectors struggle to detect all vehicles presented at all aspects. The reconciliation of the two approaches – explicit vehicle detection/classification and the location/moving-obstacle approach – seems a promising solution.

Figure 28 shows the result of explicit vehicle detection run on *Talos'* logged data. Both the *Skynet* and the DARPA chase vehicle are detected, though only in a fraction of the frames in the sequence. There were also a number of false detections that would need to be handled. Explicit vehicle detection could have possibly bootstrapped *Talos'* data association and tracking, permitting stand-off regions to be placed in front and behind *Skynet*. There still is an apparent gap in the correct treatment of active yet stationary vehicles. The posture of *Skynet* was not very different from the stationary cars parked along the side of a road (such as in “the Gauntlet” of Area B during the national qualifying event). Even with perfect vehicle detection, sensor data and modelling can only recover the current vehicle trajectory. Non-linear motions like the stop-start behaviors require conservative exclusion regions or an additional data source.

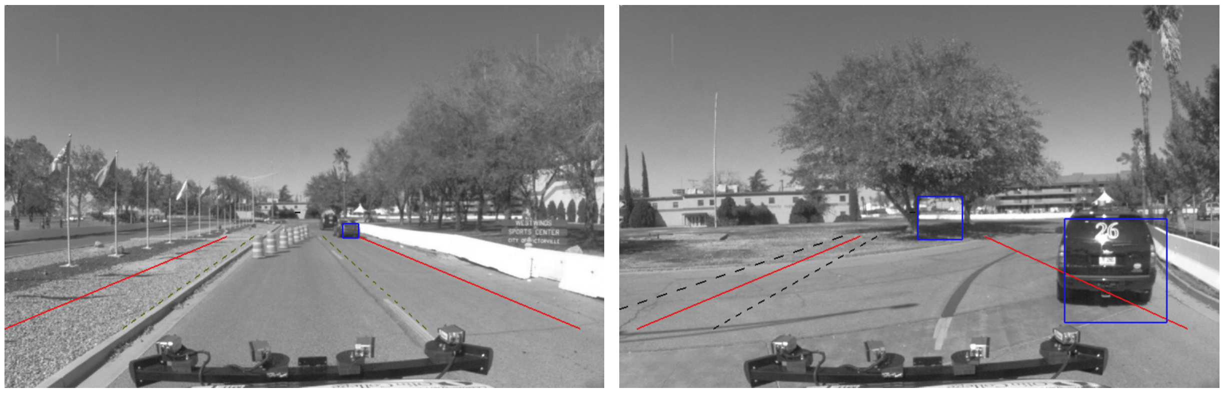

(a) (b)

Figure 28: Results of explicit vehicle detection in the collision. (a) DARPA chase vehicle detected. (b) Last frame Skynet is detected. Trees and clutter in the background also generate false positives during the sequence. In the intersection there are no lane markings so lane estimate confidence cannot be used to exclude the false detections.

8.3 Communicating intent

Drivers on the road constantly anticipate the potential actions of fellow drivers. For close maneuvering in car parks and intersections, for example, eye contact is made to ensure a shared understanding. In a debriefing after the contest, DARPA stated that traffic vehicle drivers, unnerved by being unable to make eye-contact with the robots, had resorted to watching the front wheels of the robots for an indication of their intent. As inter-vehicle communication becomes ubiquitous, autonomous vehicles will be able to transmit their intent to neighboring vehicles to implement the level of coordination beyond what human drivers currently achieve using eye-contact. This would not help in uncollaborative environments such as defense. There are also many issues such as how to handle incomplete market penetration of the communications system or inaccurate data from equipped vehicles. However, a system where very conservative assumptions regarding other vehicle behavior can be refined using the intent of other vehicles, where available, seems a reachable objective. We look forward to these synchronized robot vehicle interactions.

8.4 Placing lane constraints in context

Leading up to the collision both *Talos* and *Skynet* substantially constrained their behavior based on the lane boundaries, even though the physical world was substantially more open. *Skynet* lingered in the intersection because the lane was narrowed due to an interaction between the lane modeling and the intersection geometry. Then the vehicles collided due to a funneling effect induced by both vehicles attempting to get the optimum approach into the outgoing lane. The vehicles were tuned to get inside the lane constraints quickly; this behavior was tuned for cases such as the Area A test during the national qualifying event, in which the vehicles needed to merge into their travel lane quickly to avoid oncoming traffic. In test Area A, the robots needed to drive assertively to get into the travel lane to avoid the heavy traffic and concrete barriers lining the course. In the collision scenario, however, the road was one-way, so the imperative to avoid oncoming traffic did not exist, yet the imperative to meet the lane constraints remained. For future urban vehicles, in addition to perception, strong cues for behavior tuning are likely to come from digital map data. Meta data in digital maps is likely to include not only the lane position and number of lanes but also shoulder drivability, proximity to oncoming traffic and partition type. This *a-priori* information vetted against perception can then be used to weigh up the imperative to maximize clearance from detected obstacles with the preference to be within the lane boundaries. A key question is how the quality of this map data will be lifted to a level of assured accuracy which is sound enough to base life-critical motion planning decisions on.

9 Conclusion

The fact that the robots, despite the crash, negotiated many similarly complex situations successfully and completed the race after 6 hours of driving implied that the circumstances leading to the collision were the product of confounding assumptions across the two vehicles. Investigating the collision, we have found that bugs, the algorithms in the two vehicles architectures as well as unfortunate features of the road leading up to the intersection and the intersection topology all contributed to the collision.

Despite separate development of the two vehicle architectures, common issues can be identified. These issues reveal hard cases that extend beyond a particular software bug, vehicle design or obstacle detection algorithm. They reflect complex trade-offs and challenges: (1) Sensor data association in the face of scene complexity, noise and sensing "aperture" problems. (2) The importance of the human ability to anticipate the expected behavior of other road users. This requires an estimation of intent beyond the observable physics. Inter-vehicle communication has a good chance of surpassing driver eye-contact communication of intent, which is often used to mitigate low speed collisions. However, incomplete system penetration and denial of service for defense applications are significant impediments. (3) The competing trade-offs of conforming to lane boundary constraints (crucial for avoiding escalating problems with oncoming traffic) verses conservative obstacle avoidance in an online algorithm. Map data and meta data in maps about oncoming traffic and road shoulder drivability would be an invaluable data source for this equation. However, map data would need to be accurate enough to support safety-critical decisions.

Multimedia Appendices

Talos' race logs, log visualization software as well as videos of the incidents made from the logs are available at: http://grandchallenge.mit.edu/public/

Acknowledgments

The MIT and Cornell teams were sponsored by Defense Advanced Research Projects Agency, Program: Urban Challenge, ARPA Order No. W369/00, Program Code: DIRO.

The MIT team also gratefully acknowledges the sponsorship of: MIT School of Engineering, MIT Computer Science and Artificial Intelligence Laboratory (CSAIL), MIT Department of Aeronautics and Astronautics, MIT Department of Electrical Engineering and Computer Science, MIT

Department of Mechanical Engineering, The C. S. Draper Laboratory, Franklin W. Olin College of Engineering, The Ford-MIT Alliance, Land Rover, Quanta Computer Inc., BAE Systems, MIT Lincoln Laboratory, MIT Information Services and Technology, South Shore Tri-Town Development Corporation and Australia National University. Additional support has been provided in the form of in-kind donations and substantial discounts on equipment purchases from a variety of companies, including Nokia, Mobileye, Delphi, Applanix, Drew Technologies, and Advanced Circuits.

Team Cornell also gratefully acknowledges the sponsorship of Singapore Technologies Kinetics, Moog, Septentrio, Trimble, Ibeo, Sick, MobilEye, The Mathworks, Delphi, and Alpha Wire.

References

DARPA (2007). Darpa urban challenge rules. http://www.darpa.mil/GRANDCHALLENGE/rules.asp.

Ferguson, D., Stentz, A., and Thrun, S. (2004). Pao* for planning with hidden state. In *Proceedings of the 2004 International Conference on Robotics and Automation*, volume 3, pages 2840–2847.

Frazzoli, E., Dahleh, M. A., and Feron, E. (2002). Real-time motion planning for agile autonomous vehicles. *AIAA Journal of Guidance, Control, and Dynamics*, 25(1):116–129.

Leonard, J., How, J., Teller, S., Berger, M., Campbell, S., Fiore, G., Fletcher, L., Frazzoli, E., Huang, A., Karaman, S., Koch, O., Kuwata, Y., Moore, D., Olson, E., Peters, S., Teo, J., Truax, R., Walter, M., Barrett, D., Epstein, A., Mahelona, K., Moyer, K., Jones, T., Buckley, R., Attone, M., Galejs, R., Krishnamurthy, S., and Williams, J. (2008). A perception driven autonomous urban robot. submitted to *International Journal of Field Robotics*, (-):–.

Martin, M. and Moravec, H. (1996). Robot evidence grids. Technical Report CMU-RI-TR-96-06, The Robotics Institute, Carnegie Mellon University, Pittsburgh.

Miller, I. and Campbell, M. (2007). Rao-blackwellized particle filtering for mapping dynamic environments. In *Proceedings of the 2007 International Conference on Robotics and Automation*, pages 3862–3869.

Miller, I., Campbell, M., Huttenlocher, D., Nathan, A., Kline, F.-R., Moran, P., Zych, N., Schimpf, B., Lupashin, S., Kurdziel, M., Catlin, J., and Fujishima, H. (2008). Team cornell's skynet: Robust perception and planning in an urban environment. submitted to *International Journal of Field Robotics*, (-):–.

Russell, S. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Prentice Hall, Pearson Education, Inc., Upper Saddle River, New Jersey, 2nd edition.

Stanford Racing Team (2007). Stanford's robotic vehicle Junior: Interim report. http://www.darpa.mil/GRANDCHALLENGE/TechPapers/Stanford.pdf.

Sukthankar, R. (1997). *Situational Awareness for Tactical Driving*. PhD thesis, The Robotics Institute, Carnegie Mellon University.

Tartan Racing (2007). Tartan racing: A multi-modal approach to the DARPA urban challenge. http://www.darpa.mil/ GRANDCHALLENGE/TechPapers/Tartan_Racing.pdf.

Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V., Stang, P., Strohband, S., Dupont, C., Jendrossek, L.-E., Koelen, C., Markey, C., Rummel, C., van Niekerk, J., Jensen, E., Alessandrini, P., Bradski, G., Davies, B., Ettinger, S., Kaehler, A., Nefian, A., and Mahoney, P. (2006). Stanley: The robot that won the DARPA Grand Challenge. *J. Robot. Syst.*, 23(9).

Willemsen, P., Kearney, J. K., and Wang, H. (2003). Ribbon networks for modeling navigable paths of autonomous agents in virtual urban environments. In *Proceedings of IEEE Virtual Reality 2003*, pages 22–26.