The MIT–Cornell Collision and Why It Happened

Luke Fletcher, Seth Teller, Edwin Olson, David Moore, Yoshiaki Kuwata, Jonathan How, and John Leonard
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139
e-mail: lukesf@mit.edu

Isaac Miller, Mark Campbell, Dan Huttenlocher, Aaron Nathan, and Frank-Robert Kline
Cornell University
Ithaca, New York 14853
e-mail: itm2@cornell.edu

Received 15 February 2008; accepted 3 September 2008

Midway through the 2007 DARPA Urban Challenge, MIT's robot "Talos" and Team Cornell's robot "Skynet" collided in a low-speed accident. This accident was one of the first collisions between full-sized autonomous road vehicles. Fortunately, both vehicles went on to finish the race and the collision was thoroughly documented in the vehicle logs. This collaborative study between MIT and Cornell traces the confluence of events that preceded the collision and examines its root causes. A summary of robot–robot interactions during the race is presented. The logs from both vehicles are used to show the gulf between robot and human-driver behavior at close vehicle proximities. Contributing factors are shown to be (1) difficulties in sensor data association leading to an inability to detect slow-moving vehicles and phantom obstacles, (2) failure to anticipate vehicle intent, and (3) an overemphasis on lane constraints versus vehicle proximity in motion planning. Finally, we discuss approaches that could address these issues in future systems, such as intervehicle communication, vehicle detection, and prioritized motion planning.

1. INTRODUCTION

On November 3, 2007, the Defense Advanced Research Projects Agency (DARPA) Urban Challenge Event (UCE) was held in Victorville, California. For the first time, 11 full-size autonomous vehicles interacted with each other and human-driven 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.

On the whole, the robots drove predictably and safely through the urban road network. None of the robots stressed the (understandably) conservative

Journal of Field Robotics 25(10), 775–807 (2008) © 2008 Wiley Periodicals, Inc.
Published online in Wiley InterScience (www.interscience.wiley.com). • DOI: 10.1002/rob.20266

Figure 1. The collision. Left: Skynet. Right: Talos. This paper explores the factors that led to the collision. Despite the mishap, both vehicles went on to complete the race.

safety measures taken by DARPA. There were, however, a number of low-speed incidents during the challenge. This paper reviews those incidents and takes an in-depth look at one of them, the collision between Team Cornell's vehicle "Skynet" and MIT's "Talos" (Figure 1). 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. Several graded dirt roads were added for the competition. The contest was cast as a race against time to complete three 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 h before the race. The RNDF is very similar to a digital street map used by an in-car global positioning system (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 a list of checkpoints within the RNDF that 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 "Main Circuit" (visible in bottom left corner of Figure 2) at the end of each "submission."

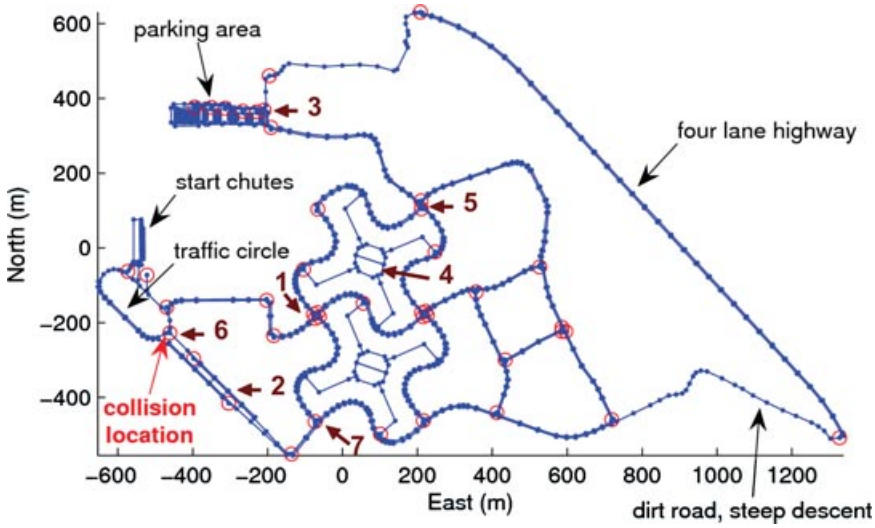


Figure 2. The UCE road network. Waypoints are designated as blue dots. Traversable lanes and zone boundaries are represented as blue lines. Stop lines are designated as red circles. The Skynet–Talos collision happened upon entering the Main Circuit on the bottom left.

Each mission was subdivided into six or seven submissions. The vehicles returned to the finishing area at the end of each mission so that 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. Additional traffic was 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-h 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 out in Sections 6 and 7 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 real urban roads.

2. CHRONOLOGY OF ROBOT–ROBOT INTERACTIONS

Table I is a list of robot–robot collisions or close calls during the UCE. The list was compiled from the race day Webcast and vehicle data logs. The locations of the incidents are marked in Figure 2. We invited teams with vehicles actively involved in the incidents [CarOLO, Intelligent Vehicle Systems (IVS), and Ben Franklin Racing Team] to coauthor or comment on the interactions. Received comments are included in the incident descriptions. 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. The University of Central Florida's (UCF's) Knight Rider was at the intersection. Skynet pulled up behind a traffic vehicle, which

Table I. Robot–robot collisions or close calls during the race.

Time (Approx.)	Location	Description	Reference
1 h 00 min	Utah and Washington	Cornell's Skynet passing with IVS's XAV-250 and Ben Franklin Racing Team's Ben oncoming	Section 2.1
1 h 30 min	George Boulevard	Ben and Team UCF's Knight Rider	Section 2.2
2 h 00 min	North Nevada and Red Zone	CarOLO's Caroline turns across MIT's Talos	Section 2.3
3 h 00 min	White Zone	Caroline and Talos collide	Section 2.4
4 h 00 min	Carolina Avenue and Texas Avenue	Talos swerves to avoid VictorTango's Odin	Section 2.5
4 h 30 min	George Boulevard and Main Circuit	Skynet and Talos collide	Section 5
5 h 20 min	Utah and Montana	Talos turns across Ben	Section 2.6

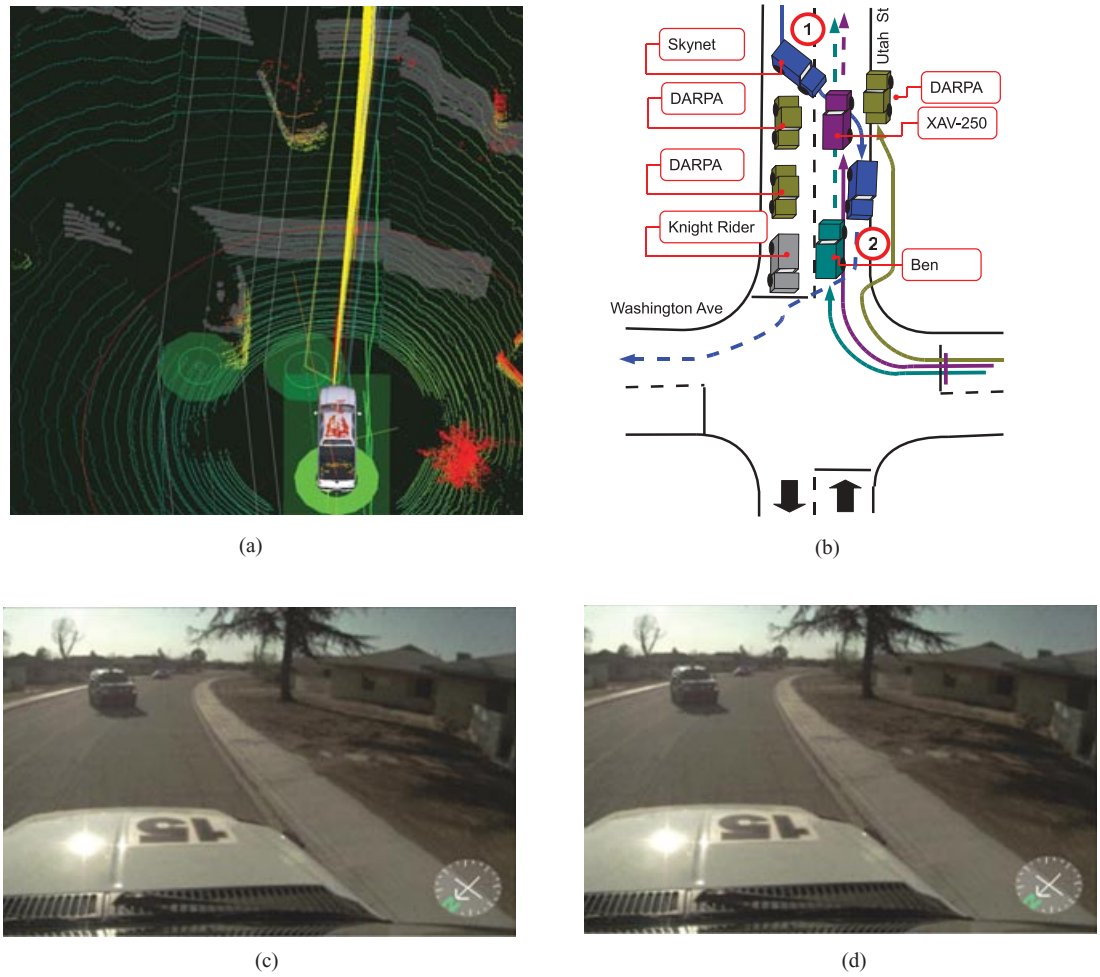


Figure 3. After queuing behind the stationary Knight Rider, Skynet passes. (a) Visualization of XAV-250 log when approaching Skynet (image courtesy of Team IVS). (b) Diagram of incident: (1) Vehicle positions when XAV-250 approaches. (2) Vehicle positions when Ben approaches. (c) XAV-250 camera view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

was queued behind Knight Rider's chase vehicle (the chase vehicle was queued behind Knight Rider). The relative positions of the 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 IVS vehicle XAV-250 turned right from Washington onto Utah and into the oncoming 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 oncoming to Skynet. Skynet was paused. Initially Ben stopped as the way was blocked by Skynet. The Skynet DARPA chase vehicle then moved to permit Ben to pass on the left. Interestingly, Ben's chase vehicle drove onto the curb around to the right to pass the Skynet vehicle. This provides an example of the

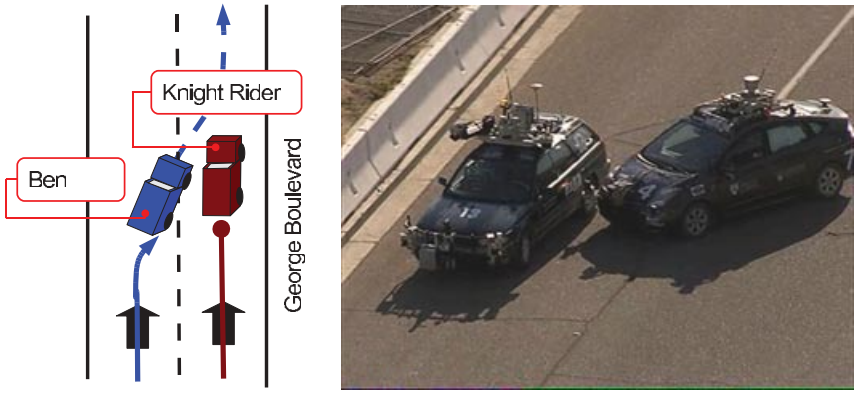


Figure 4. (a) Diagram of incident. (b) Knight Rider (13) and Ben (74) near miss.

assessment made by a human driver in this scenario. Faced with Skynet in the oncoming 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

Figure 4 shows a near miss featured in the Webcast in which Ben appeared to be merging into Knight Rider on George Boulevard. Earlier Knight Rider had merged in front of Ben. Knight Rider's chase vehicle, due to insufficient space, had been forced to follow behind Ben. The vehicles continued on to George Boulevard. The course was laid out such that on George Boulevard, a dual-lane road, vehicles began in the left lane and then were required to move to the right lane before turning at the end of the road. DARPA decided to pause Knight Rider as soon as it had moved into the right lane, expecting Ben to continue past in the left lane, permitting Knight Rider and its chase vehicle to be reunited. However, Ben, after initially checking for sufficient space, had already commenced a right merge maneuver as Knight Rider was paused. Unexpectedly, Ben found Knight Rider stopped in the destination lane. Ben was paused by DARPA in time to prevent the collision.

2.3. Caroline and Talos at North Nevada and Red Zone

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 and then turned left into the Red Zone across the path of Talos.

Talos detected the moving object of Caroline and found the closest intersection exit to project the assumed trajectory. Talos's intended motion plans were then severed by Caroline's predicted trajectory, so the vehicle commenced an emergency stop. DARPA paused both vehicles. A full account of this event from Talos's 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 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

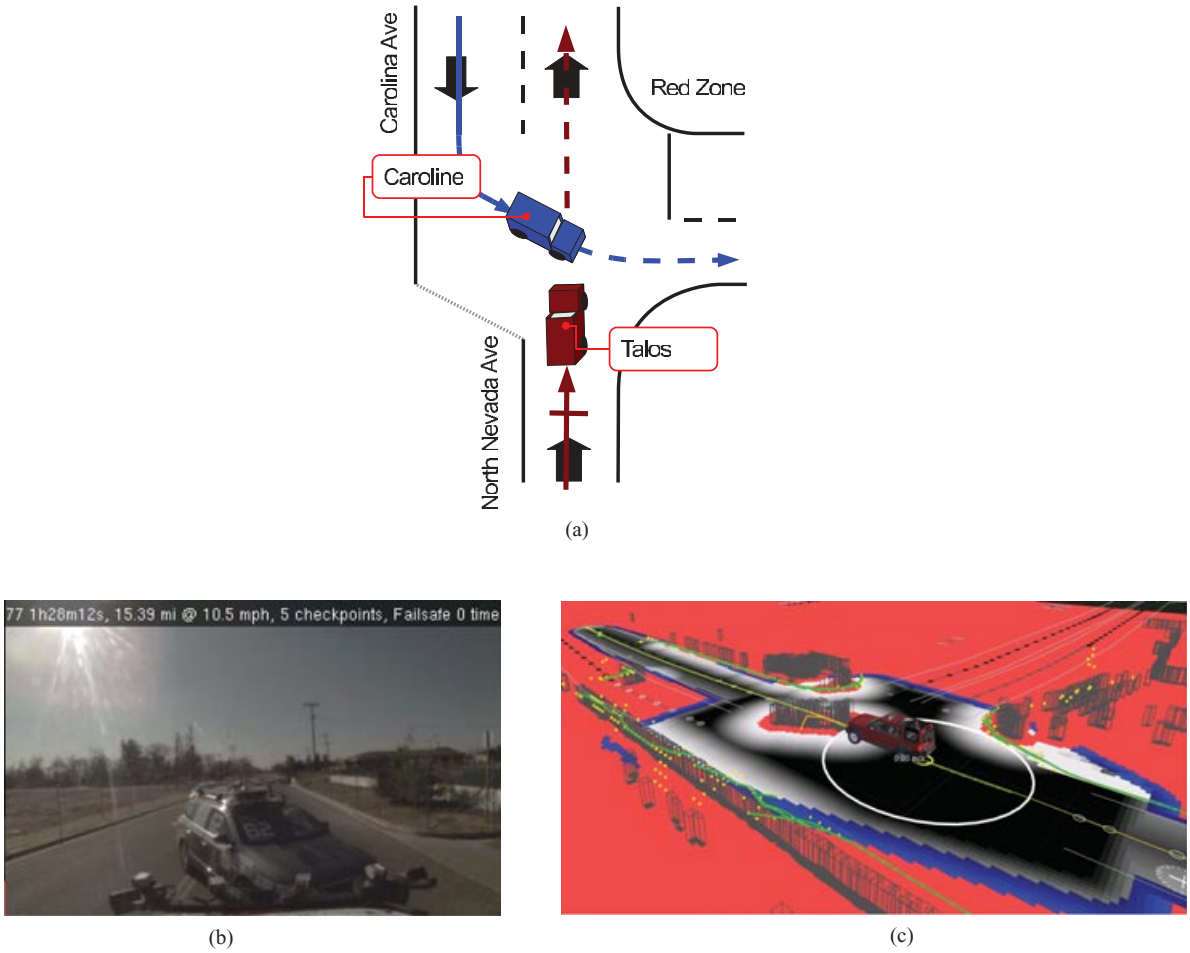
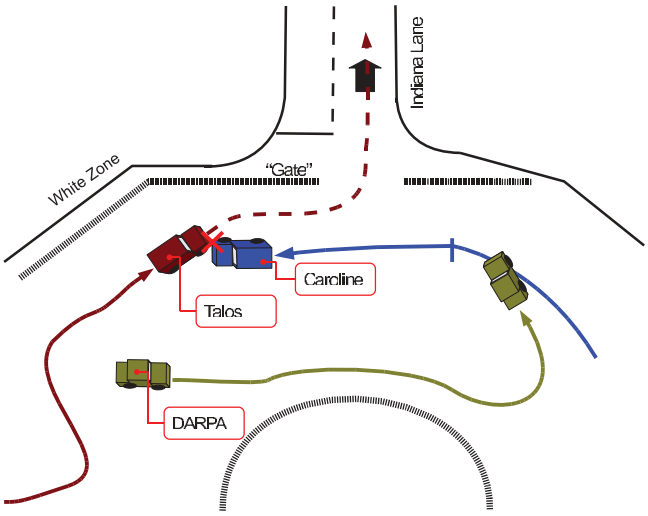


Figure 5. (a) Diagram of incident. (b) Talos's view of the final pose Caroline–Talos turning near miss. (c) Visualization from Talos's log.

exit. As Talos drove toward the zone exit, Talos also approached Caroline. Initially Caroline was stationary and then drove slowly forward toward Talos. Talos, with the zone fence to the left and what it perceived as a static obstacle (which was actually Caroline) to the right, attempted to negotiate a path in between. Caroline advanced toward Talos. Talos kept adjusting its planned path to drive around to the left of what appeared to the Talos software as a stationary object. Just before the collision Talos's motion plans were severed, causing a "planner emergency stop." Owing to Talos's 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's view is given in Leonard et al. (2008).

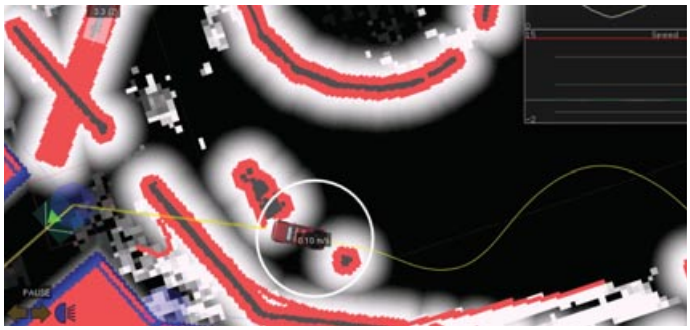
2.5. Odin and Talos at Carolina and Texas

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 arrived at a stop line at the intersection of Carolina and Texas. Talos was intending to go from Oregon to Texas [from bottom to top in Figure 7(a)]. Talos yielded to Odin approaching. Odin arrived at the intersection intending to turn left into Texas Avenue. Odin came to a stop entering the intersection. Talos detected that the time to contact for approaching vehicles had gone to infinity and so proceeded across the intersection. Odin also proceeded from Carolina Avenue into Texas Avenue. Odin, much quicker off the mark, was ahead of Talos.



(a)

(b)



(c)

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

Talos reacted to Odin approaching by braking and replanning an evasive maneuver, turning hard to the left. Odin and Odin's chase vehicle completed the turn and cleared the intersection. Talos then resumed course down Texas Avenue behind the vehicles.

2.6. Ben and Talos at Utah and Montana

The final incident, a close call, is illustrated in Figure 8. Log data show that Talos turned left from Montana onto Utah. Talos arrived at the intersection and yielded to oncoming traffic. Ben was approaching, so Talos remained stopped. At the intersection Ben also came to a stop. Again, Talos detected that the time to contact for approaching vehicles had now gone to infinity, so commenced the left-hand turn. As Talos crossed in front of Ben, Ben then also entered the intersection. At this point, Ben was quite far to the right of Talos, so Talos's forward path collision checking was not altered by the vehicle approaching to the side. Talos exited the intersection while Ben came to a stop. Once the intersection was clear, Ben continued the mission.

3. TEAM MIT'S TALOS

This section is a summary of the Talos software architecture. The purpose is to describe the vehicle

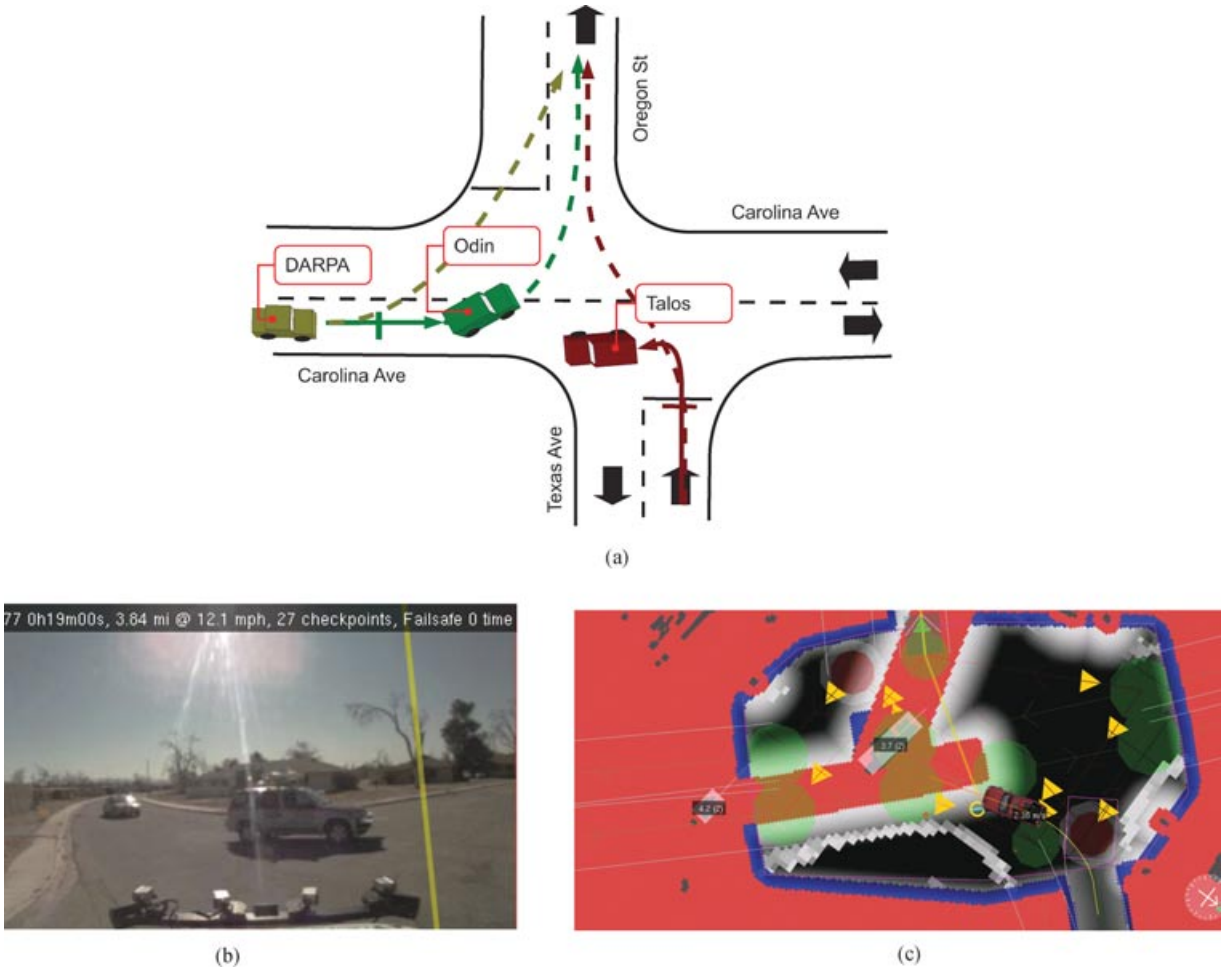


Figure 7. (a) Diagram of incident. (b) View from Talos's front camera. (c) Talos's log visualization. Odin is turning right. Talos brakes and turns hard left to avoid Odin's projected motion direction.

software in sufficient detail to understand the vehicle behavior and factors contributing to the collision. A thorough description of the robot architecture is given in Leonard et al. (2008).

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 seven horizontal SICK LMS-291 LIDARs. Five additional downward-facing SICK LMS-291 LIDARs are used for road-surface hazard detection including curb cuts. Fifteen 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 decomposes mission-level decisions into a series of short-term (1–60 m) 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 the drivability map. The

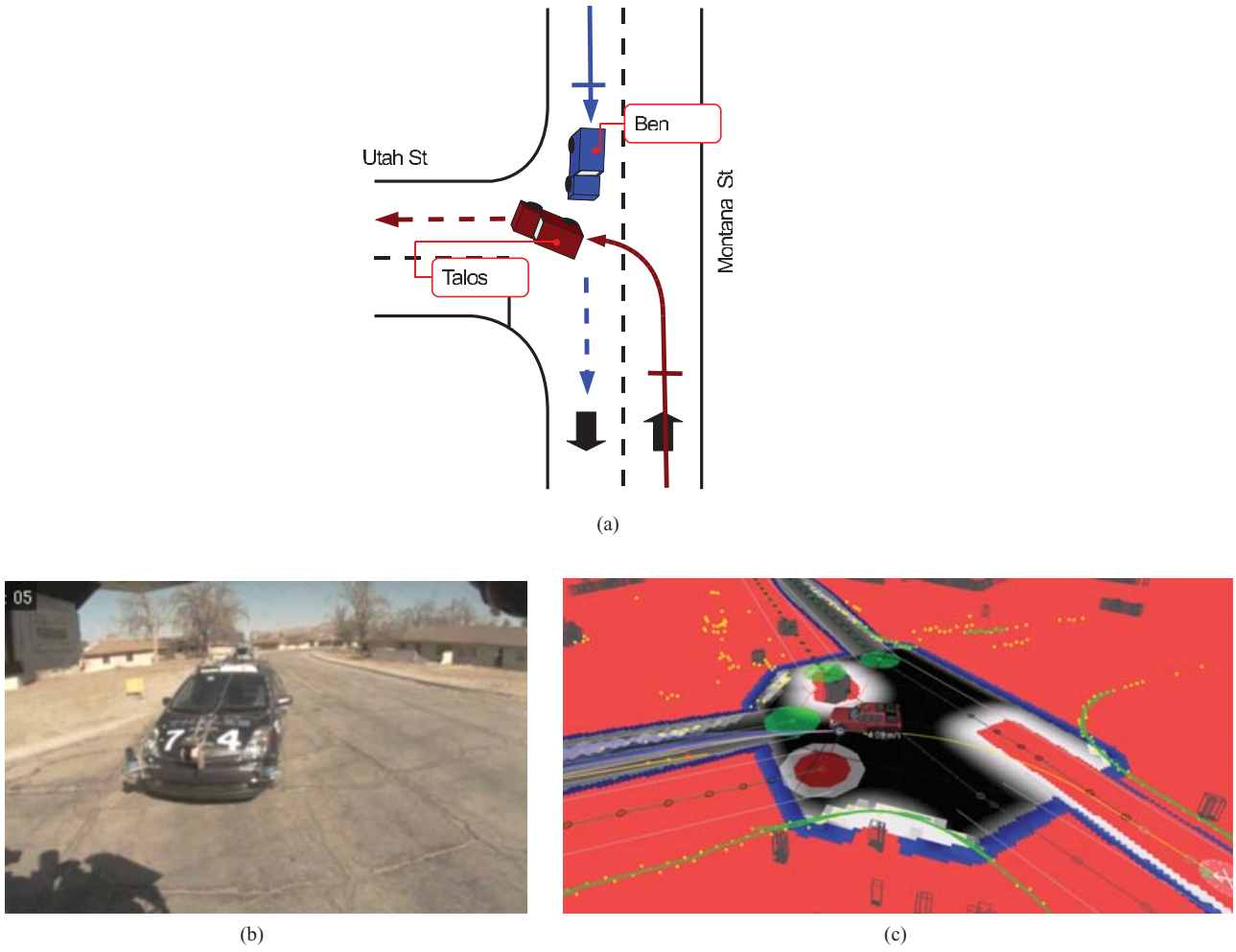


Figure 8. (a) Diagram of incident. (b) View from Talos's right front camera. Talos's view of the Little Ben–Talos turning near miss. Talos yielded to the velocity track of oncoming Ben (74). Ben came to a stop at the intersection. Talos began motion. Ben began to go through the intersection. Talos saw Ben as a creeping "static obstacle" and continued. Talos completed the turn. Ben stopped. (c) Talos's log visualization. Ben approaching on the right of Talos.

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 ensured that the autonomous vehicle could 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 s 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 centerline constraint was 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.

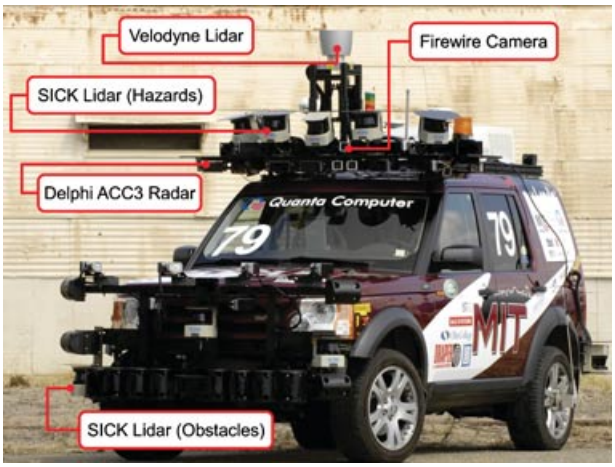


Figure 9. MIT's Talos, a Land Rover LR3 featuring five Point Grey FireFly cameras, 15 Dephi ACC3 radars, 12 SICK LMS-291 LIDARS, and a Velodyne HDL-64 LIDAR.

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 was 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 nonground detections over relatively short periods of time in an accumulator. In our implementation, the tracker ran at 15 Hz (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 system. The reliable detection range (with no false negatives) was about 30 m, with good detections out to about 60 m (but with occasional false negatives). The system was tuned to minimize false positives.

For detecting vehicles, an initial implementation simply classified any object that was approximately the size of a car as a car. In cluttered urban scenes this approach quickly led to many false positives. An alternative approach was developed based on the clustering and detection of moving objects in the scene. This approach was much more robust in cluttered environments; however, one new issue arose. In particular circumstances, stationary objects could appear to be moving. This was due to the changing viewpoint of our vehicle combined with aperture/occlusion effects of objects in the scene. Owing to this effect, a

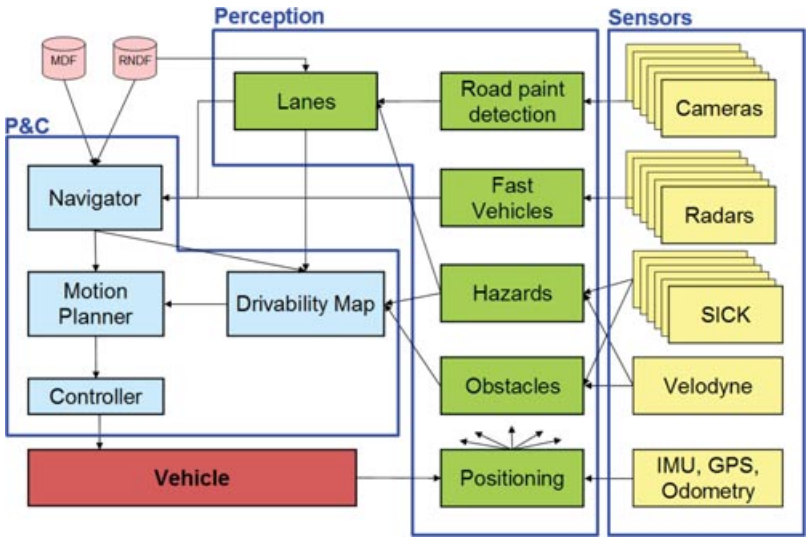


Figure 10. MIT's Talos system architecture.

high velocity threshold (3.0 m/s in our implementation) was used to reduce the frequency at which stationary objects were reported to be moving. Downstream software was written to accommodate that vehicles could appear as a collection of stationary objects or as moving obstacle tracks.

Vehicles were also detected using the radar-based fast-vehicle detector. The narrow 18-deg-field-of-view (FOV) radars were fanned to provide 255-deg coverage in front of the vehicle. The raw radar detections were compensated for vehicle egomotion and 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 consisted 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 were too small to be detected by the obstacle detector. High values in the hazard map were 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 did not 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 centerlines. The map data were 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 provided an

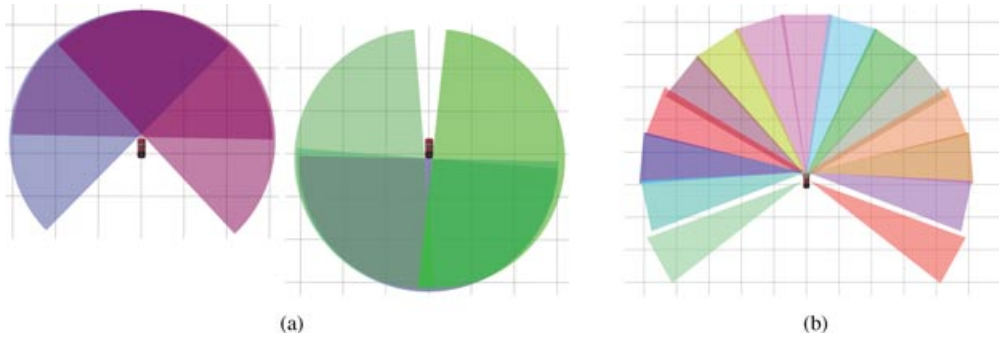


Figure 11. Sensor FOVs on 20-m grid. (a) Our vehicle used a total of seven horizontally mounted 180-deg planar LIDARs with overlapping FOVs. Front and rear LIDARs have been drawn separately to make the overlap more obvious. For ground plane rejection two LIDARs were required to "see" the same obstacle to register the detection (except in the very near field). (b) Fifteen radars with 18-deg FOV each were fanned to yield a wide (255-deg) total FOV.

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 the following:

- "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
- "restricted regions," which were regions that could be entered only if the vehicle were able to stop in an unrestricted area farther 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' standoff 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 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 kinodynamically 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, Dahleh, & Feron, 2002), where the tree of trajectories was 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 contained a measure of safety as the leaves on the tree of potential trajectories were always stopping locations [Figure 12(a)]. Shorter trees permitted lower top speeds as the vehicle had to 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 toward the goal by adding branches that connected to the randomly sampled points. These were then checked for feasibility and performance. This module then sent 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 proportional integral derivative (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 parttime contributors. Experience levels included professors, doctoral and master's candidates, undergraduates, and Cornell alumni.

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 formed the multilayer 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.

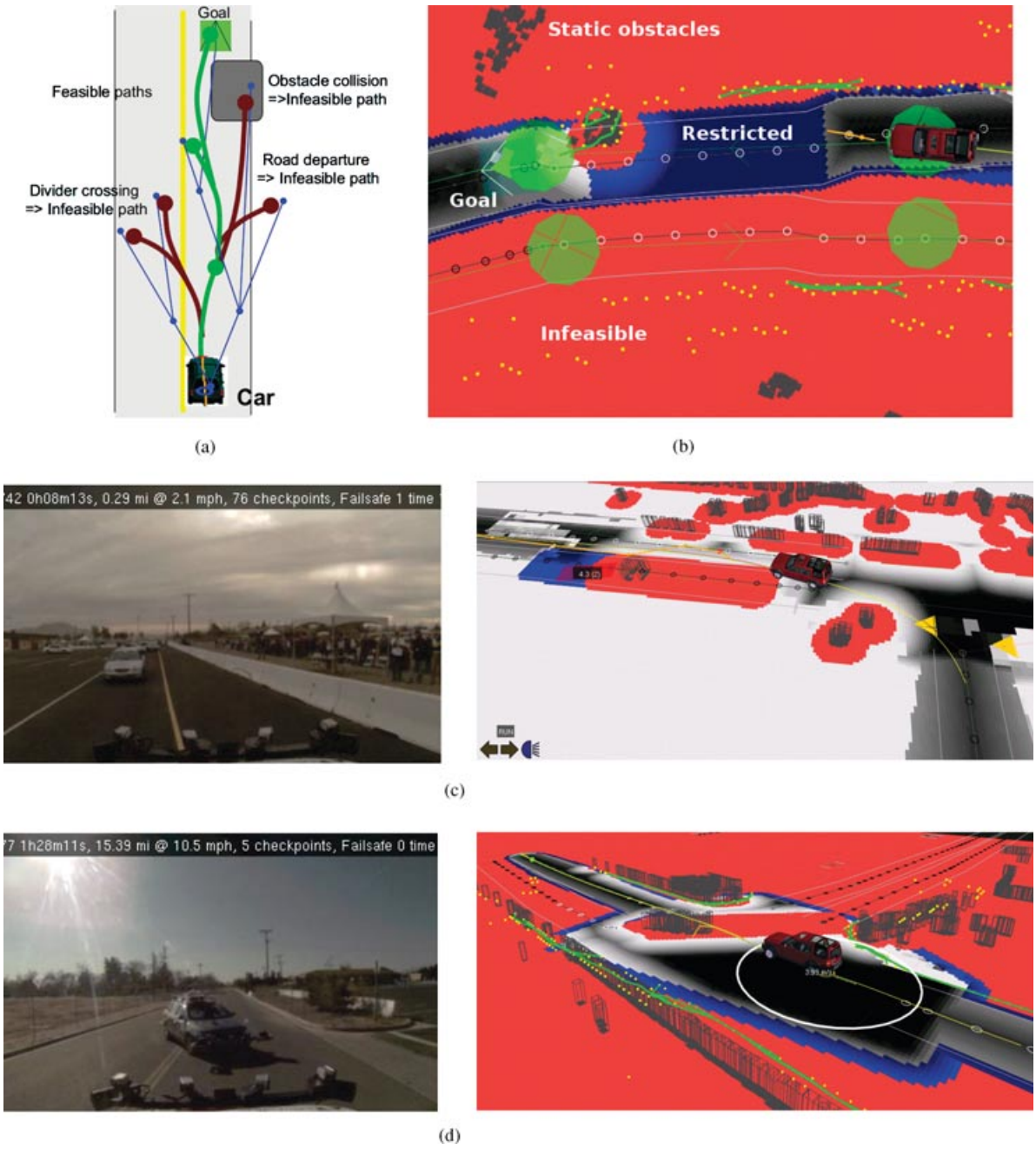


Figure 12. (a) RRT motion planning. Each leaf on the tree represented a stopping location. (b) Drivability map explanation. While driving down a two-lane road, Talos (top right) is waiting before passing a parked vehicle. Circles: Waypoints. Lines: Lanes from RNDF shifted to match detected lane markings. Arrow (left of image): Short-term goal location. Light regions (adjacent to road): Infeasible regions off-limits to the vehicle. Dark region (in front of Talos): Restricted region, which may be entered if the vehicle is able to stop in an unrestricted region farther on. Light gray shading: High-cost regions accessible to the vehicle. Black areas: Low-cost drivable regions. (c) An infeasible region was projected down lane in the direction of the moving obstacle's velocity excluding maneuvers in front of an oncoming vehicle. In this case the areas adjacent to the lane were rendered as high cost instead of infeasible due to a recovery mode triggered by the lack of progress through the intersection. (d) Within an intersection an infeasible region was created between a moving obstacle and the intersection exit matching the detected velocity direction, preventing Talos from driving into the path of the turning vehicle.

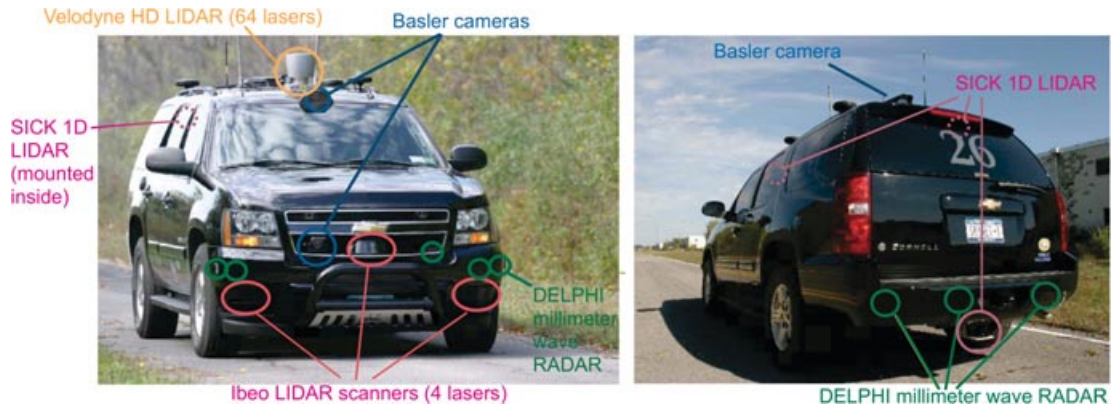


Figure 13. Team Cornell's Skynet.

4.1. General System Architecture

Skynet observed the world with two groups of sensors. Skynet's position, velocity, and attitude were sensed with raw measurements collected from GPS receivers, an inertial measurement unit (IMU), and wheel encoders. These raw measurements were fused in the 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

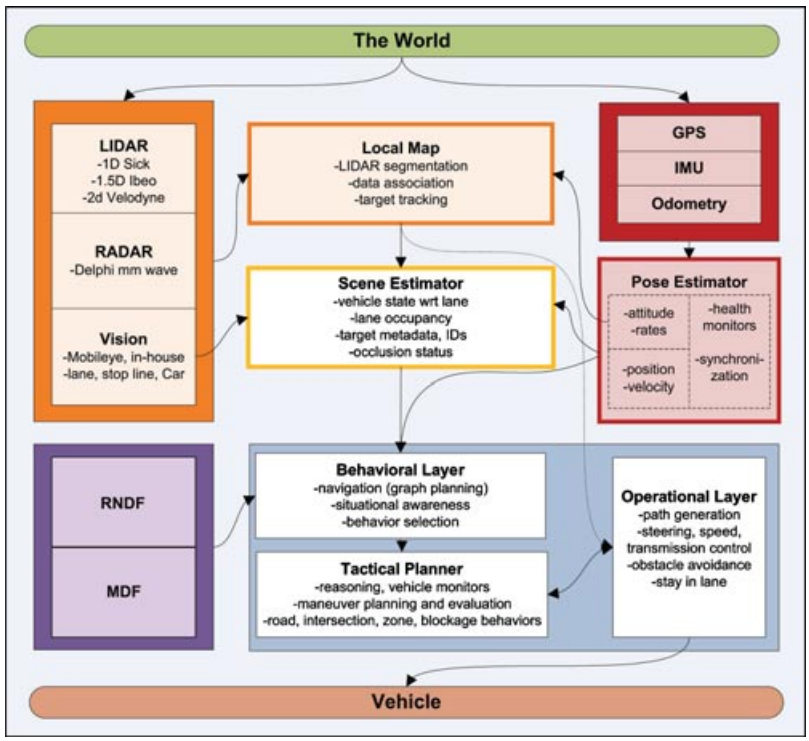


Figure 14. System architecture of Team Cornell's Skynet.

attributes of the road itself, was sensed using a combination of laser range finders, radar, and optical cameras.

Skynet used two levels of probabilistic data fusion to understand its external environment. The local map fused 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 used 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 were 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 was currently occluded.

Planning over DARPA's RNDF and 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 occurred. 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 drove the target path by converting it to a series of desired speeds and curvatures, which were 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, fused the output of all obstacle detection sensors into one vehicle-centric map of Skynet's environment. The local map fused information from three sensing modalities: laser range finders, radars, and optical cameras. Mounting positions are shown in Figure 13. Table II summarizes Skynet's obstacle detection sensors, and Figure 15 gives a top-down view of Skynet's sensor coverage. All sensor measurements were 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) fitted easily into this framework, as their proprietary algorithms transmitted lists of tracked obstacles. Data from the laser range finders were clustered to fit into this object-level framework.

The local map formulated obstacle detection and tracking as the task of simultaneously tracking multiple obstacles and determining which sensor measurements corresponded to those obstacles (Miller & Campbell, 2007; Miller et al., 2008). The problem was cast in the Bayesian framework of estimating a joint

Table II. Skynet's obstacle detection sensors.

Sensor	Location	Type	Rate (Hz)	FOV (deg)	Resolution
Ibeo ALASCA XT	Front bumper left	Laser	12.5	150	1 deg
	Front bumper center	Laser	12.5	150	1 deg
	Front bumper right	Laser	12.5	150	1 deg
SICK LMS 291	Left back door	Laser	75	90	0.5 deg
	Right back door	Laser	75	90	0.5 deg
SICK LMS 220	Back bumper center	Laser	37.5	180	1 deg
Velodyne HDL-64E	Roof center	Laser	15	360	0.7 deg
Delphi FLR	Front bumper left (2x)	Radar	10	15	20 tracks
	Front bumper center	Radar	10	15	20 tracks
	Front bumper right (2x)	Radar	10	15	20 tracks
	Back bumper left	Radar	10	15	20 tracks
	Back bumper center	Radar	10	15	20 tracks
	Back bumper right	Radar	10	15	20 tracks
Unibrain Fire-i 520b	Back roof center	Optical	15	20–30	N/A

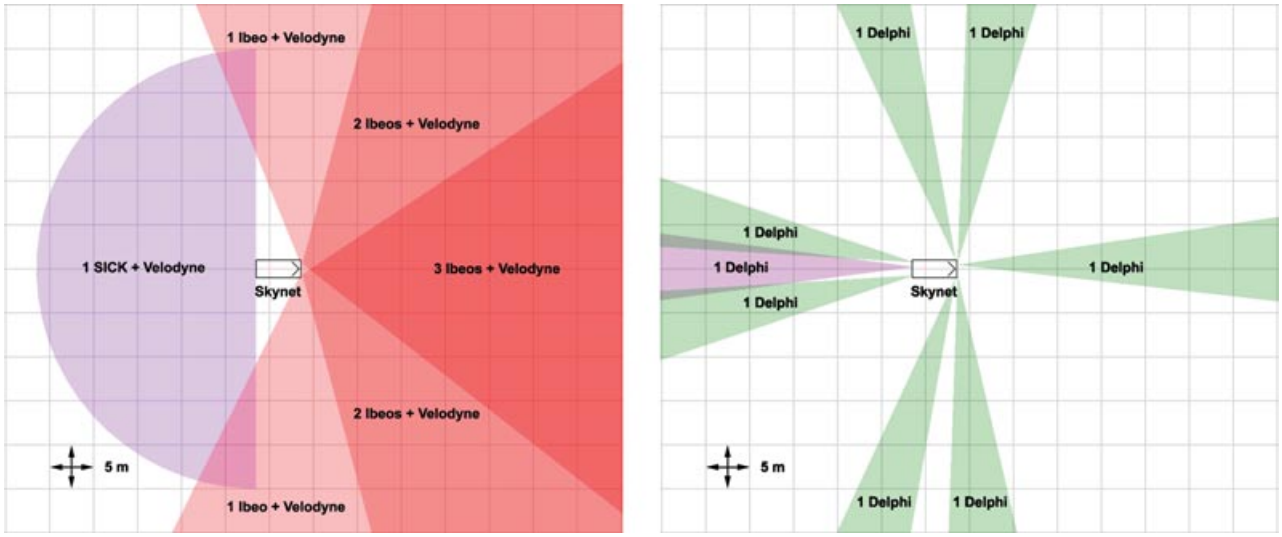


Figure 15. (Left) Laser range finder azimuthal coverage diagram for Team Cornell's Skynet. (Right) Radar azimuthal coverage diagram. Skynet faced right in both coverage diagrams. A rear-facing optical camera is not shown, nor are two laser range finders with vertical scan planes that detected obstacles immediately to the left and right of Skynet.

probability density:

$$p[N(1:k), X(1:k)|Z(1:k)], \quad (1)$$

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

$$p[N(1:k)|Z(1:k)] \cdot p[X(1:k)|N(1:k), Z(1:k)], \quad (2)$$

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 were 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 & Campbell, 2007; Miller et al., 2008). The obstacles were then broadcast at 10 Hz on Skynet's data network. A second layer, called the track generator, combined these obstacles with Skynet's position estimates to generate high-level obstacle metadata for the planner, including a stable identification number, whether each obstacle was stopped or shaped like a car, and whether each obstacle occupied any nearby lanes.

4.3. Intelligent Planning

Team Cornell's intelligent planning system used 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 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 was the most abstract layer in Team Cornell's planner. Its job was 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, was solved using a modified version of the A* graph search algorithm (Ferguson, Stentz, & Thrun, 2004; Russell & Norvig, 2003). First, the DARPA road network was converted from the RNDF format to a graphical hierarchy of segments (Willemsen, Kearney, & Wang, 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 transitioned to a new behavior state, a corresponding tactical component was executed. All components divided the area surrounding Skynet into regions and created monitors to detect events that might have influenced 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 lay in the types of region monitors they used and in the actions they took in response to nearby events.

The first tactical component was the road tactical, which controlled Skynet when it drove down an unblocked road. This component was 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 was the intersection tactical, which controlled 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 determined that Skynet was allowed to proceed, a target speed, goal point, and polygon defining the intersection were passed along to the operational layer as a reference path.

The third tactical component was the zone tactical, which controlled Skynet after it entered 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 was the blockage tactical, which controlled Skynet when obstacles blocked forward progress on the current route. This component was responsible for detecting and recovering from roadblocks 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 failed, planning constraints were relaxed: first the admissible lane boundaries were widened, and then obstacles were progressively ignored in order of increasing size. Skynet's recovery process escalated over several minutes in a gradual attempt to return to normal driving.

4.3.3. Operational Layer

The operational layer converted 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 processed each obstacle into a planar convex hull. The obstacles were then intersected with lane boundaries to form a vehicle-fixed occupancy grid (Martin & Moravec, 1996). The A* search algorithm was used to plan an initial path through the free portion of the occupancy grid (Russell & Norvig, 2003). This initial path was then used to seed a nonlinear trajectory optimization algorithm for path smoothing.

Skynet's nonlinear trajectory optimization algorithm attempted to smooth the initial path to one that was physically drivable, subject to actuator constraints and obstacle avoidance. The algorithm discretized 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 attempted 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 $v_i, i \in \{1, n\}$ were 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 violated desired spacings on the left and right of Skynet created from the list of polygonal obstacles. Search weights, velocities, and final obstacle spacings were chosen to minimize the cost function J:

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

where $\alpha_c, \alpha_d, \alpha_w, \alpha_q, \alpha_a$, and α_v are tuning weights; c_i is the approximated curvature at the ith path point; w_i^t is the target search weight at the ith path point; and a_i is the approximated forward vehicle acceleration at the ith path point. This cost function is optimized subject to a set of six 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.

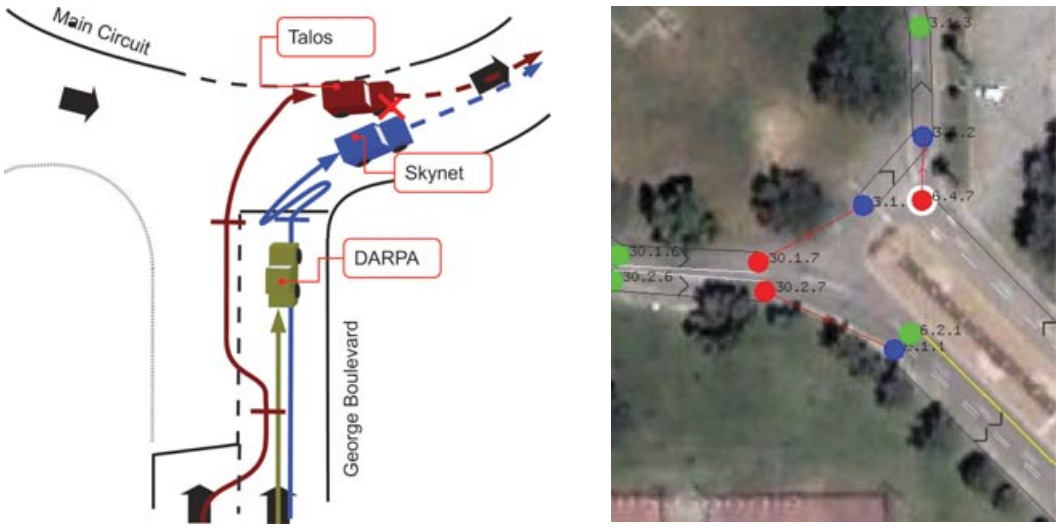
Additional constraints on initial and final path heading were also occasionally included to restrict the smoothed path to a particular end orientation, such as remaining parallel to a lane or a parking spot.

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

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 onto Main Circuit to complete their latest submission. Skynet drove down George Boulevard and was the first to arrive at the intersection. The vehicle paused, moved forward onto Main Circuit (around two car lengths), and then came to a stop. It backed up about three car lengths,



(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).

stopped, drove forward a car length, and 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 alongside 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 to 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 worked best when the lane was straight or had shallow curves: sharp turns could yield polygons that excluded significant portions of the lane. The turn at the southern entrance to the traffic circle suffered from this problem acutely, as the turn was closely bounded on the right by concrete

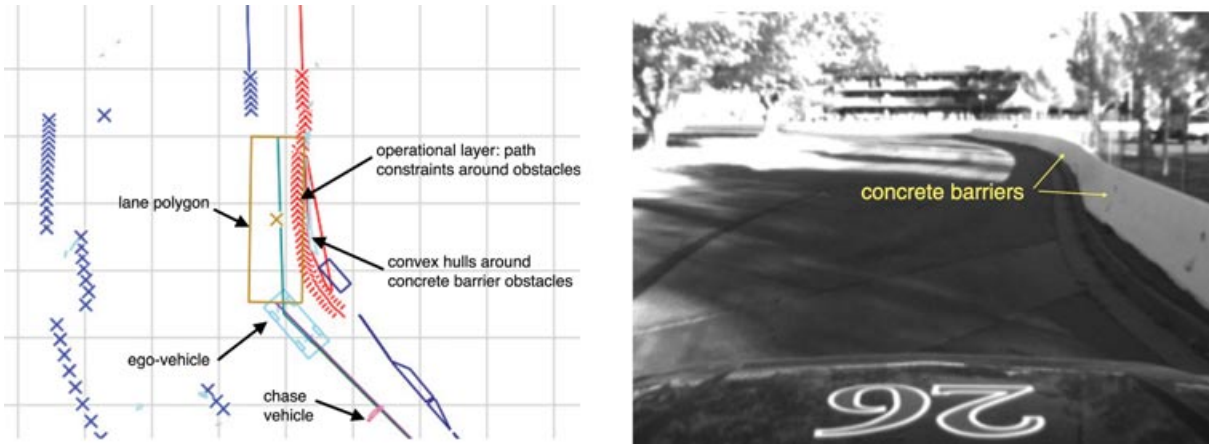


Figure 17. (Left) The lane polygon implied by piecewise-linear interpolation of DARPA waypoints in the turn near the south entrance to the traffic circle. Obstacle constraints from nearby concrete barriers occupied a significant portion of the lane polygon. (Right) Skynet camera view of the concrete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

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 could 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 range finder 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, cleared in one or two planning cycles: for these, Skynet tapped 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

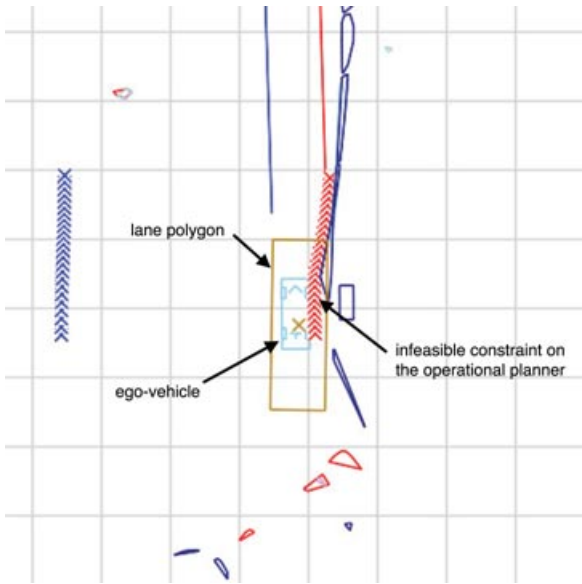


Figure 18. Small variations in Skynet's perception of a concrete barrier cause its planned path to become infeasible.

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 range finders and Delphi radars tended to produce more false detections when

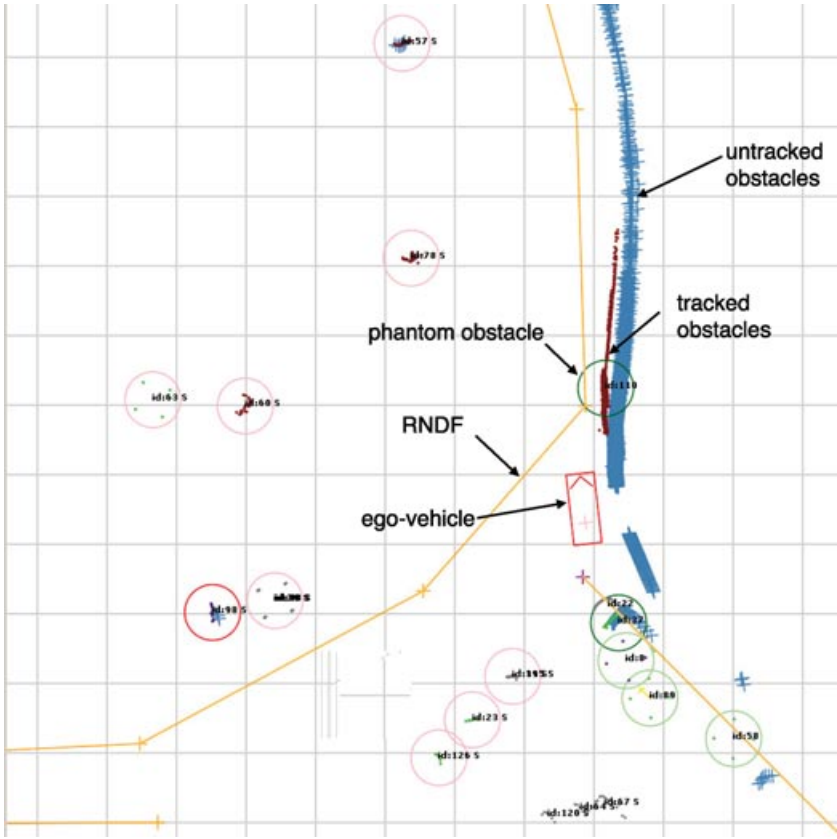


Figure 19. A measurement assignment mistake causes a phantom obstacle to appear, momentarily blocking Skynet's path.

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 were 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 did not have enough supporting sensor evidence to delete the phantom obstacle immediately and allowed it to persist until that evidence was accumulated. When this happened, Skynet considered the path blocked and executed the blockage recovery tactical component to deal with the situation. Blockage recovery was activated 10 times over the 6 h of the UCE.

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 partway 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 s, 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 that 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

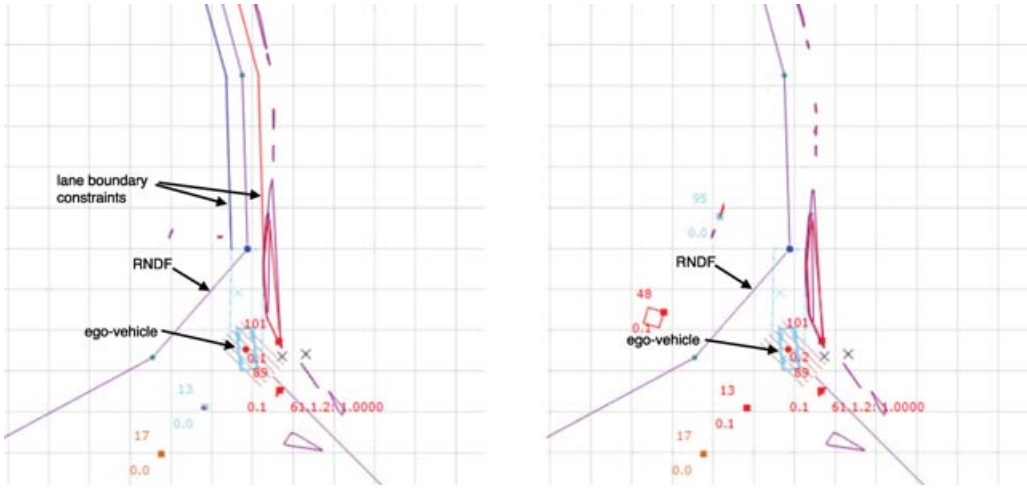


Figure 20. (Left) Skynet resumed its turn after a reverse maneuver. (Right) Perceiving the turn infeasible a second time, Skynet dropped constraints associated with lane boundaries.

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 that 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 range finder 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.

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, just as Talos pulled out to pass Skynet on the left, a slight variation in the obstacle clustering and measurement assignments accumulated enough evidence in the local map to perceive the path as feasible. 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 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 were mounted with a vertical scan plane and provided only weak position information and no velocity information. The local map began tracking Talos as a moving obstacle only 1 s 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, given the preceding chain of events, the collision was inevitable.

Figure 22 shows the speed and heading, as estimated on Skynet, for both the Skynet and Talos vehicles. Skynet tracked Talos's approach until approximately 0.5 s before collision, where Talos was too close to be sensed correctly. After that, Skynet assumed Talos continued to drive at its prior speed and heading. Skynet did not change its heading or

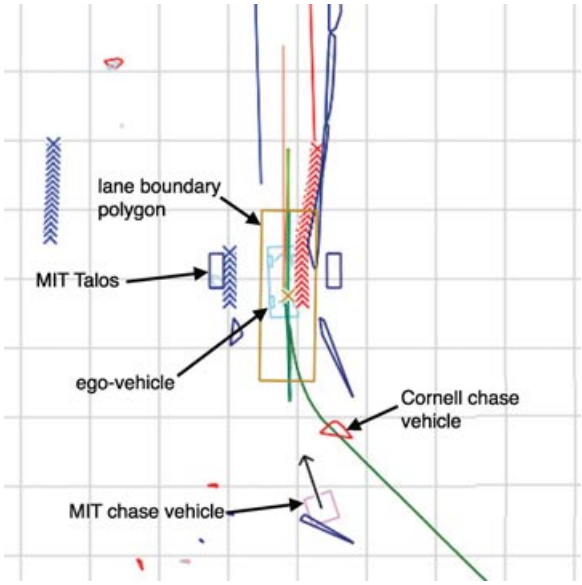


Figure 21. Skynet ignored Talos as it drove outside Skynet's polygonal lane boundary.

velocity before the collision, indicating that no adjustments were made to the Talos movements. Finally, after impact, there was 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 23–25. 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 came to a stop at the intersection [Figure 23(c)].

In Figure 24 we see that at first, Talos stopped behind the chase vehicle. However, the lane width was sufficient that Talos soon found a path to the left of the chase vehicle [Figure 24(a)]. In this case Talos was not in a passing mode; it had 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 was significantly wider to the left because of a drivability map construction bug. As described in Section 3, lanes were carved out of the lane-cost map

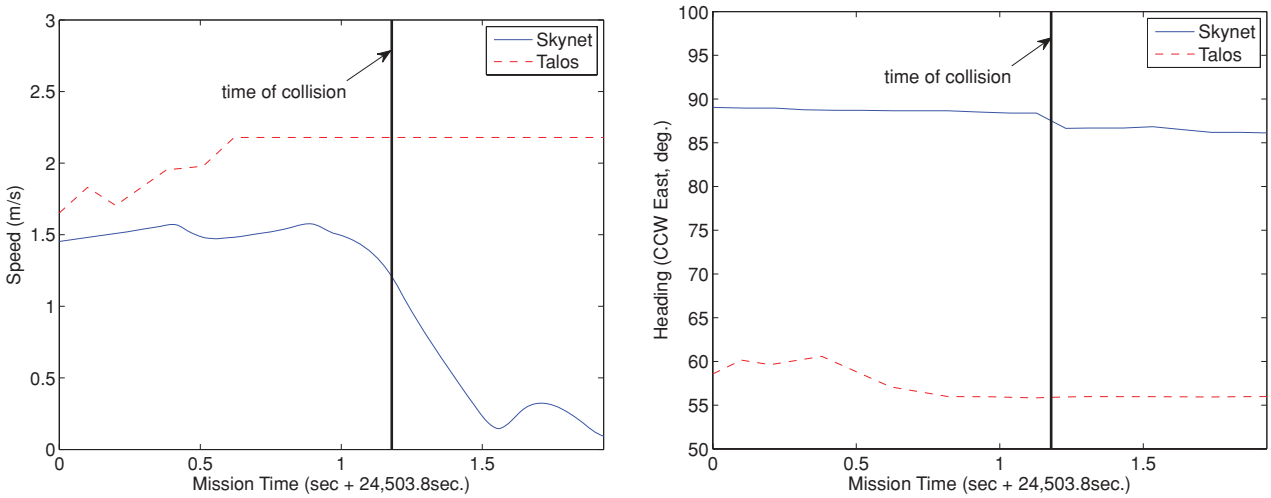


Figure 22. From Skynet logs: Speed (left) and heading (right) for both Skynet and Talos just before and after collision. Flat line in Talos's plot indicates where Skynet stopped tracking Talos.

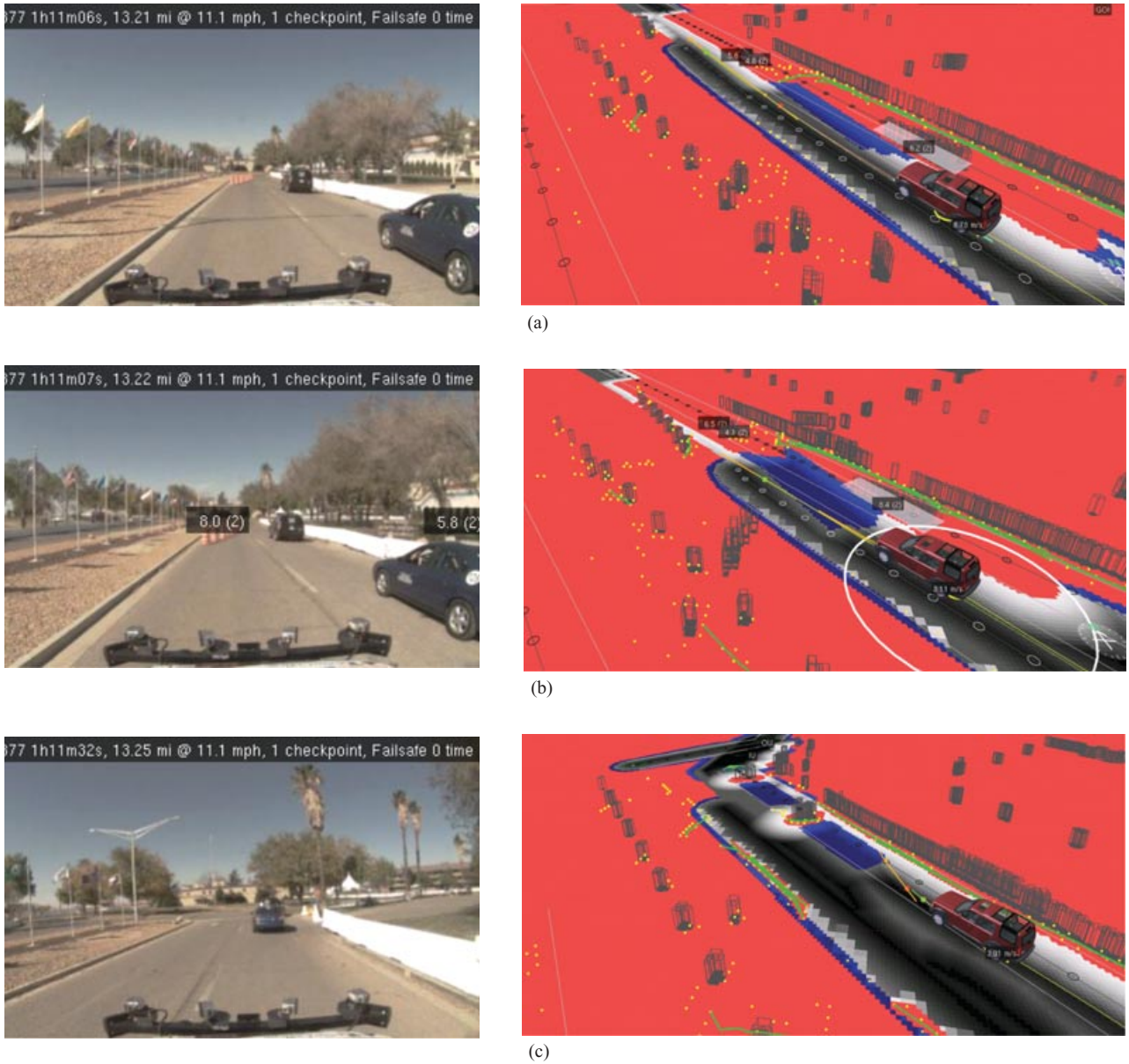


Figure 23. Talos's view of the lead-up to the Skynet–Talos incident. (a) Talos started to pass Skynet's chase vehicle. (b) Talos was forced to slow down and merge behind the Skynet chase vehicle. (c) Talos queued behind the Skynet chase vehicle.

like valleys through a plateau. Adjacent lanes carved out often resulted in 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 merged down to one lane shortly before the intersection. The adjacent lane was not rendered after the merge, which was correct. However, the planing operation was done all the way along the right lane past the merge point. The effect of the planing alone made the road 3 m wider on the left than it would otherwise have been. Without the extra width, Talos would have been forced to queue behind the DARPA chase vehicle.

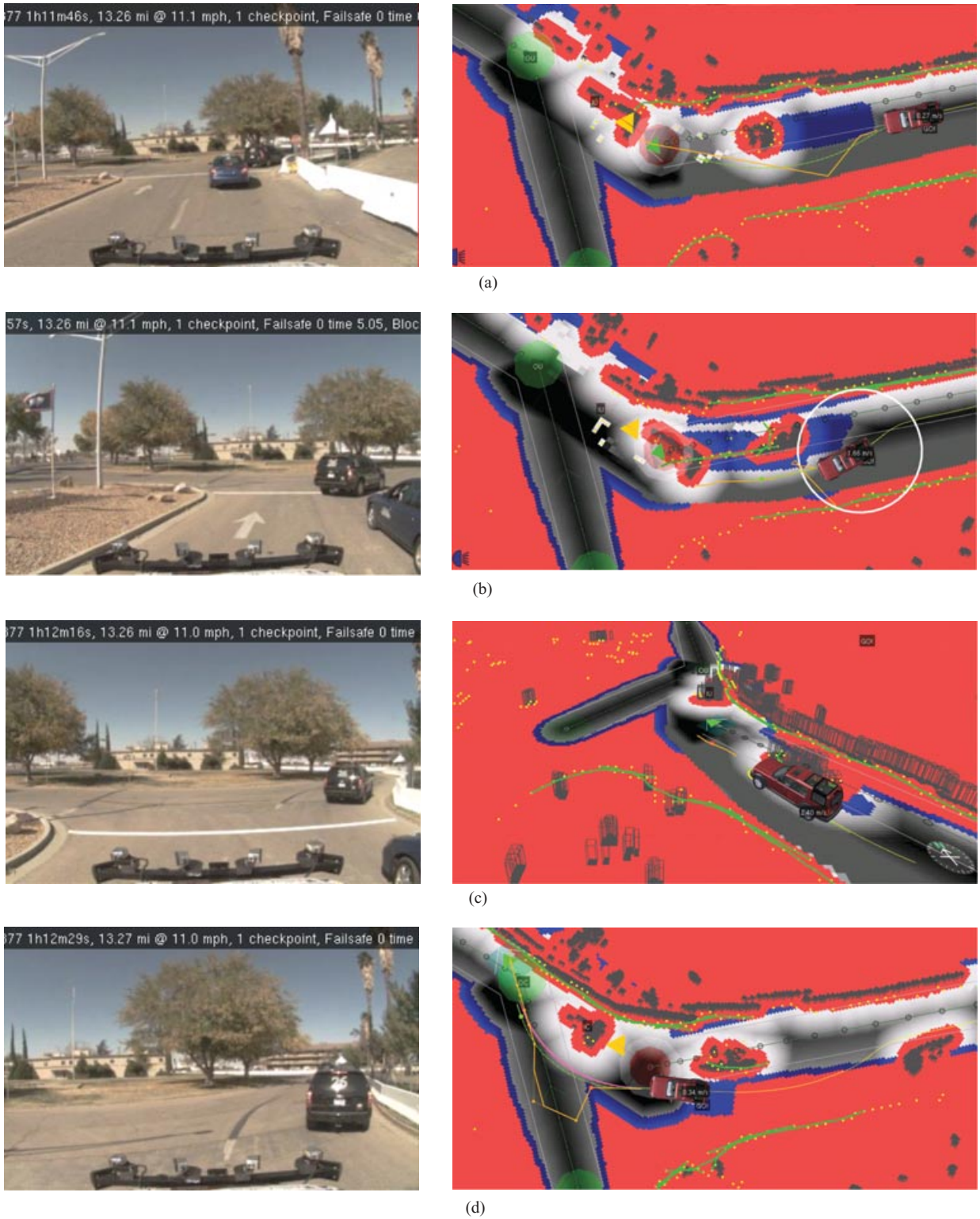


Figure 24. Lead-up to the Skynet–Talos incident. (a) Talos found a route around the chase vehicle. (b) Skynet backed up onto Talos's goal position; Talos braked. (c) Skynet advanced again. Talos passed the chase vehicle. (d) Talos yielded at the intersection. There were no moving vehicles nearby, so it proceeded.

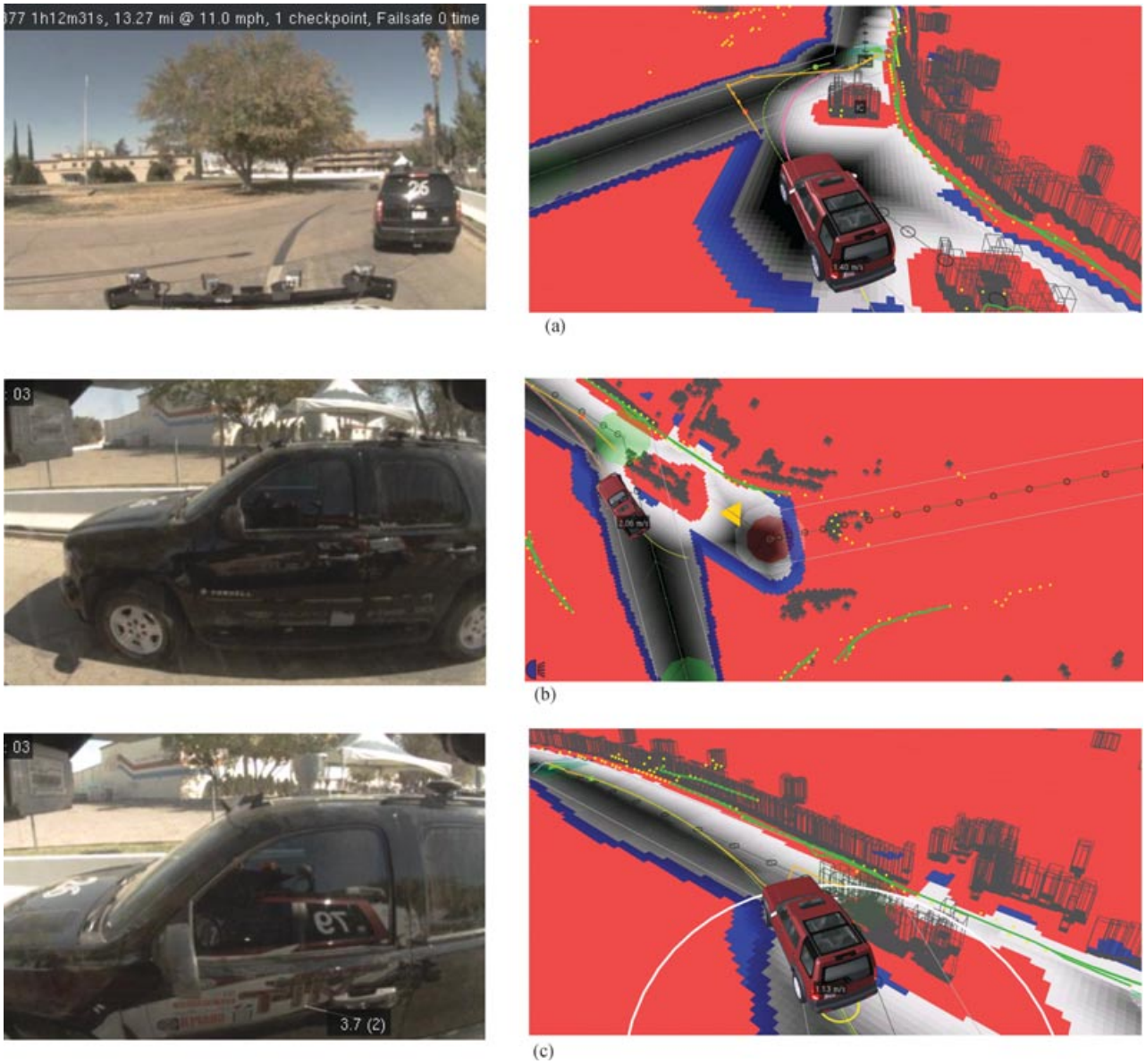


Figure 25. Skynet–Talos incident. (a) Talos planned a route around Skynet, which appeared as a static object. (b) While Talos was passing, Skynet began to move. (c) While applying emergency braking, Talos turned into Skynet.

7.2. At the Intersection

Figures 24(b) and 24(c) show how Talos pulled out and drove around to the left of the chase vehicle. The robot had a motion plan that 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's 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 of 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's goal position clear. Talos drove to the stop line. Although now adjacent to Skynet's chase vehicle, Talos was not in a fail-safe mode. The artificially widened lane permitted Talos to drive up to the intersection as it would passing a 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 were 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 that Talos planned 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's goal point moved farther 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., >3 m/s instead of 1.5 m/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 as described in Section 2.5). Instead Talos turned to the right to get the correct approach angle to drive down the lane; Skynet moved forward; the robots collided [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, for instance, to 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)–23(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, 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's path, Talos was unable to come to a stop before colliding.

Figure 26 shows the vehicle state during the collision. Talos had straightened its wheels to around 9 deg to the right and was traveling around 2 m/s. The vehicle detected that the motion planning tree had been severed 550 ms before the collision. It was replanning, and no evasive maneuver was performed yet; 150 ms later the vehicle was coasting and DARPA paused the vehicle. At the collision Talos was moving at 1.5 m/s, dropping to zero 750 ms after the initial collision. In the log visualization Talos was pushed slightly forward and to the left of its heading by the impact (about 0.3 m).

The contributing factors of Talos's behavior can be decomposed as the inability to track slow-moving objects, the use of a moving-obstacle model versus explicit vehicle classification, and the dominant influence of lane constraints on motion planning and 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

At the lowest layer, all objects tracked by the obstacle detection system had a velocity component. However, both sensor noise and changing viewpoint geometry can masquerade as small velocities, making it difficult to reliably determine whether an object is actually moving. The problem of changing viewpoint is

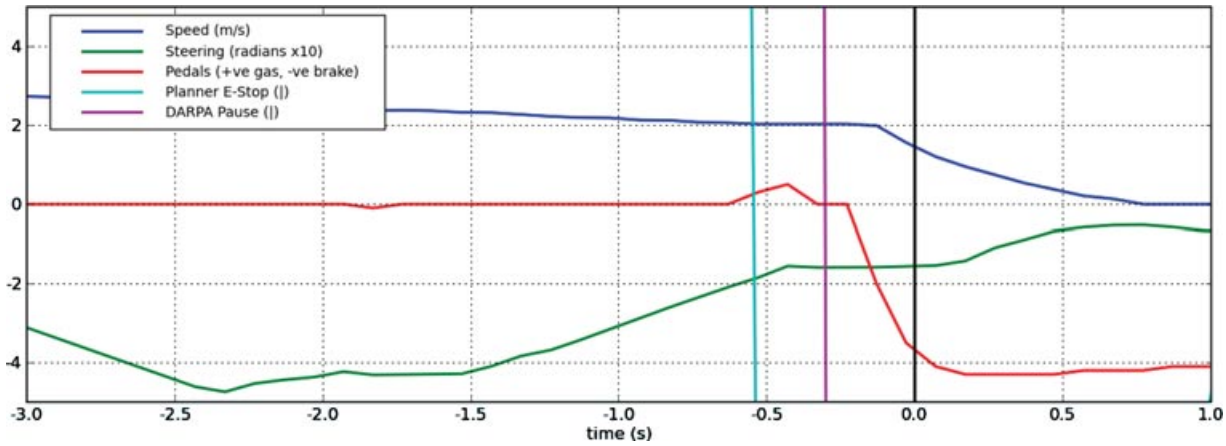


Figure 26. Talos's speed, wheel angle, and pedal gas and brake positions during the collision. Time 0.0 is the initial collision. Motion planner E-stop was at −550 ms. DARPA pause at −400 ms. The vehicle came to rest after 750 ms.

especially problematic. If Talos is moving, the visible portion of other obstacles can change; the "motion" of the visible portion of the obstacle is difficult to distinguish from an obstacle that is actually moving.

Apertures between the sensor and the obstacle present additional complications. Figure 27 shows an example in which a small near-field aperture resulted in a hallucinated car. In this case, only a small patch

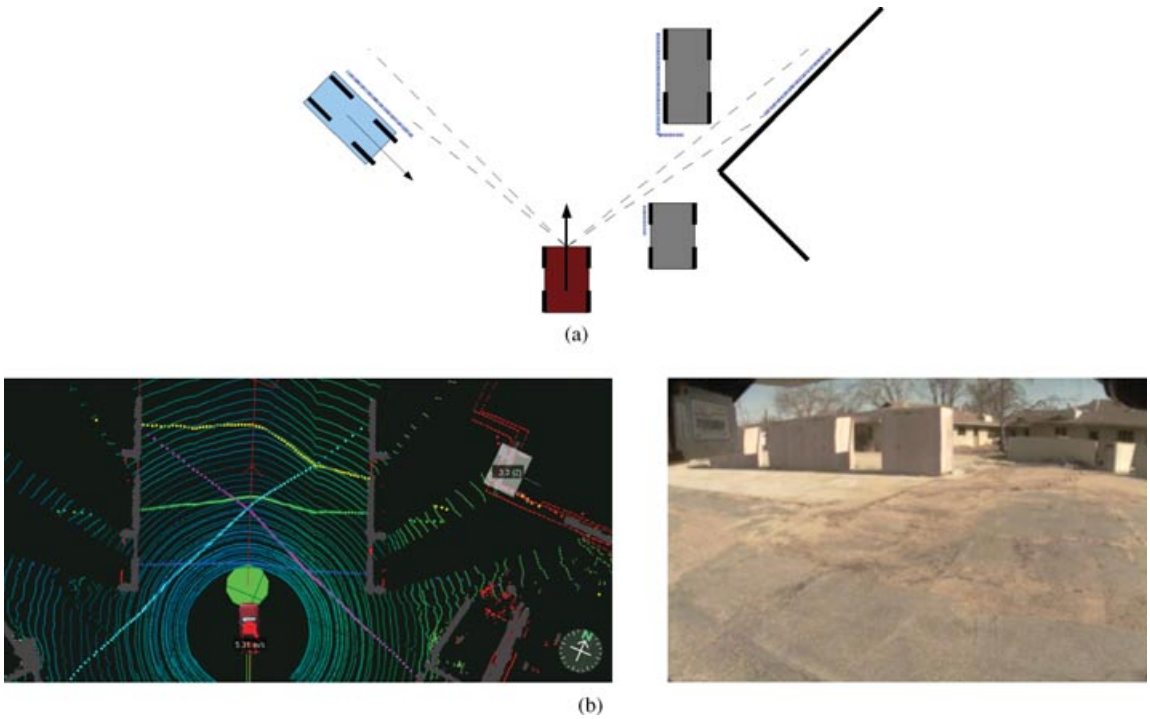


Figure 27. (a) Illustration of LIDAR aperture problem. The building on the right generates a LIDAR return indistinguishable from the fast-approaching vehicle on the left. (b) Walls entering the White Zone generate phantom moving obstacles from building LIDAR returns.

of wall was visible through the aperture: as Talos moved, an object appeared to be moving in the opposite direction on the other side of the aperture. 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. A flat obstacle occlusion map is no longer sufficient because obstacle height must be considered.

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 as either 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 "nonvehicle" classes. The integrated system was fragile, however, as classification errors or outages caused failures in downstream 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. The rationale for this was that sensor data classification schemes, by requiring a choice to be made, introduced the potential for false-positive and false-negative classifications. Much care could be taken in reducing the likelihood of misclassifications; however, classification errors could almost always still occur. Developers have often designed downstream applications to be overconfident in the classes assigned to objects. Avoiding the assignment of "vehicle"/"nonvehicle" classes to detected objects was an attempt to cut down assumptions made by the downstream 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 was not very different from that of 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 ended in stopped states, so that Talos always knew how to come to a safe stop. When Talos was moving and the planner could not 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 changed the constraints), the planner generated an emergency braking plan. This emergency plan consisted 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 toward the target farther down the road. When the gap between the left lane boundary and Skynet was narrowed as 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 and then pulled in much too close to an active vehicle in an intersection.

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

- Talos's lane-rendering bug permitting Talos to pass the DARPA chase vehicle
- Talos's inability to track slow-moving objects

- Skynet's sensor data associations inducing phantom objects
- Talos's 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's overly constrained motion due to target lane constraints
- Skynet's lane representation narrowing the drivable corridor

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

8.1. Sensor Data Clustering

Skynet's phantom obstacles and Talos's 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 that until the threshold was passed, there was no motion prediction for slow-moving objects.

8.2. Implicit and Explicit Vehicle Detection

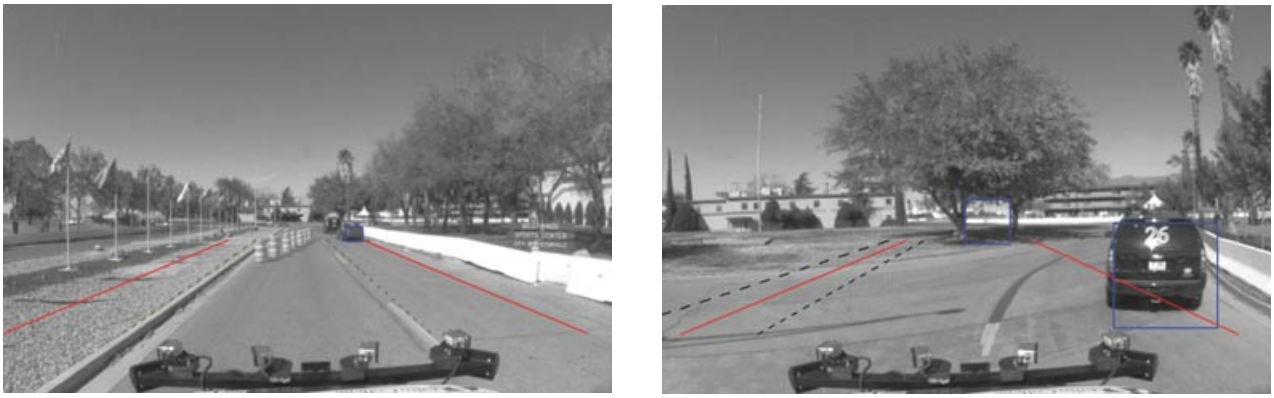
The treatment of vehicles in the road environment must extend past the physics-based justification of obstacle avoidance due to closing rate. For example, humans prefer never to drive into the longitudinal path of an occupied vehicle, even if it is stationary. In Section 2 we mentioned how the DARPA chase vehicle driver preferred to drive on the curb rather 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 in which 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's logged data. Both Skynet and the DARPA chase vehicle are detected, though in only 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's data association and tracking, permitting standoff regions to be placed in front of and behind Skynet. There still was an apparent gap in the correct treatment of active yet stationary vehicles. The posture of Skynet was not very different from those of the stationary cars parked along the side of a road [such as in the gauntlet of Area B during the National Qualifying Event (NQE)]. Even with perfect vehicle detection, sensor data and modeling can recover only the current vehicle trajectory. Nonlinear motions like the stop–start behaviors require conservative exclusion regions or an additional data source.

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 intervehicle communication becomes ubiquitous, autonomous vehicles will be able to transmit their intent to neighboring vehicles to implement the level of coordination beyond



(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.

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 in which 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 NQE, 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. 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. Metadata in digital maps are 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 could 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 the map data will be lifted to a level of assured accuracy that 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 h 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 and 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. Intervehicle 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) versus conservative obstacle avoidance in an online algorithm. Map data and metadata 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's race logs and log visualization software, as well as videos of the incidents made from the logs, are available at http://grandchallenge.mit.edu/public/.

ACKNOWLEDGMENTS

The authors would like to thank all the members of their respective teams: from Team MIT Mitch Berger, Stefan Campbell, Gaston Fiore, Emilio Frazzoli, Albert Huang, Sertac Karaman, Olivier Koch, Steve Peters, Justin Teo, Robert Truax, Matthew Walter, David Barrett, Alexander Epstein, Keoni Maheloni, Katy Moyer, Troy Jones, Ryan Buckley, Matthew Antone, Robert Galejs, Siddhartha Krishnamurthy, and Jonathan Williams; from Team Cornell Jason Catlin, Filip Chelarescu, Ephrahim Garcia, Hikaru Fujishima, Mike Kurdziel, Sergei Lupashin, Pete Moran, Daniel Pollack, Mark Psiaki, Max Rietmann, Brian Schimpf, Bart Selman, Adam Shapiro, Philipp Unterbrunner, Jason Wong, and Noah Zych. In addition, the authors would like to thank Jim McBride (IVS), Matt Rupp (IVS), and Daniel Lee (Ben Franklin), who helped provide information for the events described in this paper.

The MIT team gratefully acknowledges the sponsorship of the 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.

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

REFERENCES

DARPA (2007). DARPA Urban Challenge rules. http://www.darpa.mil/GRANDCHALLENGE/rules.asp.

Ferguson, D., Stentz, A., & Thrun, S. (2004). Pao* for planning with hidden state. In Proceedings of the 2004 International Conference on Robotics and Automation, New Orleans, LA (Vol. 3, pp. 2840–2847). IEEE.

Frazzoli, E., Dahleh, M. A., & Feron, E. (2002). Real-time motion planning for agile autonomous vehicles. 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., & Williams, J. (2008). A perception driven autonomous urban robot. Journal of Field Robotics, 25(10), 727–774.

Martin, M., & Moravec, H. (1996). Robot evidence grids (Tech. Rep. CMU-RI-TR-96-06). Pittsburgh, PA: Robotics Institute, Carnegie Mellon University.

Miller, I., & Campbell, M. (2007). Rao-Blackwellized particle filtering for mapping dynamic environments. In Proceedings of the 2007 International Conference on Robotics and Automation, Rome, Italy (pp. 3862–3869). IEEE.

Miller, I., Campbell, M., Huttenlocher, D., Nathan, A., Kline, F.-R., Moran, P., Zych, N., Schimpf, B., Lupashin, S., Kurdziel, M., Catlin, J., & Fujishima, H. (2008). Team Cornell's Skynet: Robust perception and planning in

an urban environment. Journal of Field Robotics, 25(8), 493–527.

Russell, S., & Norvig, P. (2003). Artificial intelligence: A modern approach (2nd ed.). Upper Saddle River, NJ: Prentice Hall, Pearson Education, Inc.

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, Pittsburgh, PA.

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., & Mahoney, P. (2006). Stanley: The robot that won the DARPA Grand Challenge. Journal of Field Robotics, 23(9), 661–692.

Willemsen, P., Kearney, J. K., & Wang, H. (2003). Ribbon networks for modeling navigable paths of autonomous agents in virtual urban environments. In Proceedings of IEEE Virtual Reality 2003, Los Angeles, CA (pp. 79–86). IEEE.