Tartan Racing: A Multi-Modal Approach to the DARPA Urban Challenge

April 13, 2007

Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, John Dolan, Dave Duggins, Dave Ferguson, Tugrul Galatali, Chris Geyer, Michele Gittleman, Sam Harbaugh, Martial Hebert, Tom Howard, Alonzo Kelly, David Kohanbash, Maxim Likhachev, Nick Miller, Kevin Peterson, Raj Rajkumar, Paul Rybski, Bryan Salesky, Sebastian Scherer, Young Woo-Seo, Reid Simmons, Sanjiv Singh, Jarrod Snider, Anthony Stentz, William "Red" Whittaker, and Jason Ziglar

Carnegie Mellon University

Hong Bae, Bakhtiar Litkouhi, Jim Nickolaou, Varsha Sadekar, and Shuqing Zeng

General Motors

Joshua Struble and Michael Taylor

Caterpillar

Michael Darms

Continental AG

Corresponding Author:

Chris Urmson (curmson@ri.cmu.edu)

Executive Summary

The Urban Challenge represents a technological leap beyond the previous Grand Challenges. The challenge encompasses three primary behaviors: driving on roads, handling intersections and maneuvering in zones. In implementing urban driving we have decomposed the problem into five components. Mission Planning determines an efficient route through an urban network of roads. A behavioral layer executes the route through the environment, adapting to local traffic and exceptional situations as necessary. A motion planning layer safeguards the robot by considering the feasible trajectories available, and selecting the best option. Perception combines data from lidar, radar and vision systems to estimate the location of other vehicles, static obstacles and the shape of the road. Finally, the robot is a mechatronic system engineered to provide the power, sensing and mobility necessary to navigate an urban course.

Rigorous component and system testing evaluates progress using standardized tests. Observations from these experiments shape the design of subsequent development spirals and enable the rapid detection and correction of bugs. The system described in the paper exhibits a majority of the basic navigation and traffic skills required for the Urban Challenge. From these building blocks more advanced capabilities will quickly develop.

Table of Contents

Introduction and Overview

Reliable autonomous urban driving will be a revolutionary technology. It will enable uncrashable cars, driver assists, and enable robotic applications in both industrial and military sectors. The Urban Challenge represents a bold step towards fully capable autonomous urban vehicles.

At the highest level of autonomy, the problem is to find an efficient route through an urban road network with prescribed waypoints and parking spots. The three primary implementation skills are; driving in traffic, negotiating intersections and parking. Each represents a formidable reasoning, motion-planning and perceptual problem. The driving in traffic component alone represents a superset of previous Grand Challenge capability.

Motion planning must account for the kinematic and dynamic limitations of the vehicle, while constantly acting to avoid collisions and unsafe driving. On roads, the shape and width of the road constrain the planning problem, simplifying the search for viable trajectories. In parking lots, a more extensive search is necessitated to deal with both the freedom of motion and tight pose constraints necessitated by parking. Correctly solving these problem requires two distinct planning modes.

Perception must have sufficient fidelity and field of view to determine the location of other vehicles, roads and unexpected obstacles, all in a cluttered environment. No single sensing mode is sufficient to accomplish all of these requirements, so a combination of lidar, radar and visual sensing is necessitated.

To meet the challenge in the short time frame, we are using a spiral development process. Each spiral innovates a new capabilities. Each spiral matures and tests solid functionality. At the beginning of each spiral, the system is analyzed and new design decisions and tradeoffs are made. This technical paper represents much of the current conceptual design for each subsystem, current performance and a number of tradeoffs and analyses that were made during development.

Theory of Autonomous Operation

The task of autonomous urban driving can be decomposed into five major blocks (see Figure 1):

Mission Planning – The Mission Planning component computes the cost (as a function of time and risk) of all possible routes through the world given knowledge of the road network from the RNDF and the next checkpoint that the vehicle must achieve. The mission planner reasons about the optimal path to a particular checkpoint much like a human would plan a route from their current position to a destination, such as a grocery store or gas station. The mission planner compares routes based on prior knowledge of congestion or blockages, construction, and the legal speed limit.

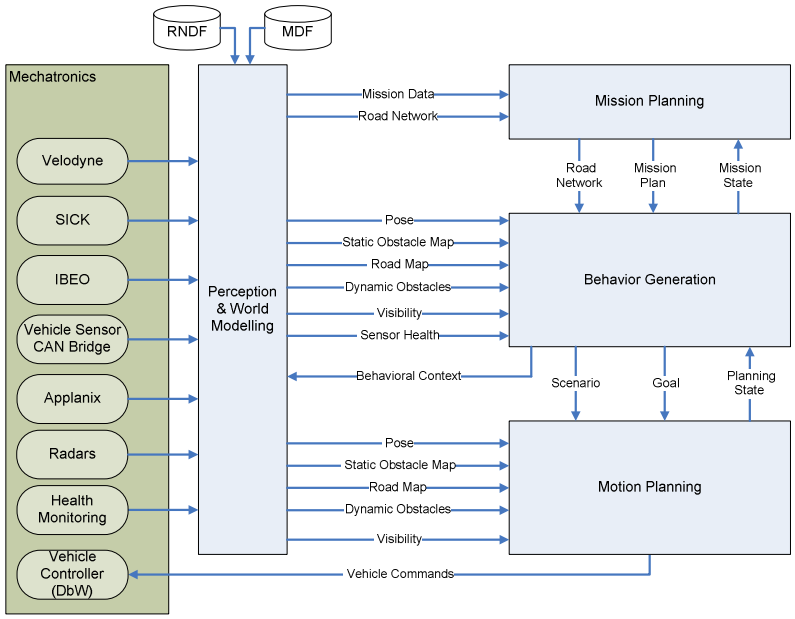


Figure 1. The Tartan Racing architecture is decompsed into five broad areas: Mission Planning, Motion Planning, Behavior Generation, Perception and World Modeling, and Mechatronics.

Behavior Generation – The Behaviors component formulates a problem definition for the Motion Planning component to solve based on the strategic information provided by the Mission Planning component. The Behaviors component is implemented as a state machine that decomposes the mission task into a set of top-level behaviors and their simpler, sub-behaviors in order to complete a mission. These top-level behaviors include *Drive-Down-Road*, *Handle-Intersection*, and *Achieve-Zone-Pose*. The state machine will trigger different behaviors based on the progress being reported by the motion planners, overall health of the various vehicle and software systems, and the current objectives set forth by the mission planner.

Motion Planning – The Motion Planning component consists of several motion planners, each capable of avoiding static and dynamic obstacles while achieving a desired vehicle state. Two broad scenarios are considered: structured driving (road following) and unstructured driving (parking lot). For structured driving, a Local Planner is capable of generating a trajectory that will result in the vehicle following the centerline of a lane. For unstructured driving, such as entering/exiting a parking lot, a 3D (position and orientation) planner is used. Regardless of the planner that is executing, the result is a trajectory that when executed by the vehicle controller, will safely drive toward a particular goal point.

Perception & World Modeling – The P&WM component interprets the information from various sensors and fuses the multiple streams together to provide a composite picture of the world to the rest of the system. The composite model is divided into several discrete interfaces: static obstacle maps, dynamic obstacle maps, visibility, health, current vehicle pose, and road structure.

Mechatronics – The electrical and mechanical components provide a way for algorithms to interact with the world. A critical part of the mechatronic design is the analysis of sensing requirements and sensor capabilities. The robotic vehicle, Boss, shown in Figure 2 is designed to be robust and provide the power, cooling and actuation necessary to support autonomy.

In the following sections each of these areas will be addressed in greater detail. Each section

includes a discussion of the design and performance results to date. The Algorithms and Mechatronics sections are followed by a section describing our testing process and overall system performance to date.

Figure 2. Boss, the Tartan Racing robot, is built on a Chevrolet Tahoe chassis. It incorporates a variety of lidar, radar and visual sensors to safely navigate urban environments.

Algorithms for Autonomous Urban Driving

There are three layers of continuous planning in the software architecture beginning with the Mission Planning, then Behavior Generation, and finally the Motion Planning. Each of these layers makes decisions based on the best information available from the Perception subsystem. The Mission Planner is reasoning about the best path from each waypoint to the next checkpoint. The Behavior Generation subsystem *initiates* inter-lane, inter-road, and parking-lot maneuvers, while the Motion Planner *executes* these maneuvers and *initiates* intra-lane maneuvers, such as in-lane obstacle avoidance.

Mission Planning

To generate mission plans, the data provided in the Route Network Definition File (RNDF) is used to create a graph that encodes the connectivity of the environment. Each waypoint in the RNDF becomes a node in this graph, and directional edges (representing lanes) are inserted between a given waypoint and all other waypoints that it can reach. For instance, an exit waypoint at an intersection will have edges connecting it to all the entry waypoints at the intersection that could be legally reached by a vehicle positioned at the exit waypoint. These edges are also assigned costs based on a combination of several factors, including expected time to traverse the edge, length of the edge, and complexity of the local environment. Behavior Generation uses this cost graph in conjunction with local, dynamic information to make decisions about travel roads and lanes.

The cost graph is searched to compute a minimum-cost path from each position in the graph to a desired goal position, such as the first checkpoint in the mission. In addition to providing the executive more information to reason about, computing minimum-cost paths from every position in the graph is useful because it allows the navigation system to behave correctly should the

vehicle be unable to perfectly execute the original plan (e.g. if a particular intersection is passed through by mistake, we can immediately extract the current best path from the vehicle's position).

As the vehicle navigates through the environment, the mission planner updates its graph to incorporate newly-observed information, such as road blockages or previously-unknown intersections or roads. Each time a change is observed, the mission planner re-generates a new plan. Because the size of the graph is relatively small, this replanning can be performed extremely quickly, allowing for immediate response to environmental changes.

Behavior Generation

The behavioral design is based on the concept of identifying driving contexts, where each context requires the vehicle software to focus on a reduced set of environmental features. At the highest level of our current design, the three contexts are *Drive Down Road*, *Handle Intersection*, and *Achieve Zone Pose* (e.g., Parking). The name "Achieve Zone Pose" stems from its coverage of sub-behaviors in unstructured or unconstrained environments, This is not specific just to parking lots, but can be used to U-turn and navigating a jammed intersection.

The *Drive Down Road* behavior (Figure 3) is responsible for on road driving. The primary sub-behavior is *Drive In Lane*, which includes a distance-keeping behavior in the presence of a lead vehicle. The Lane Selector makes lane-change decisions based on a tradeoff between making timely progress in the current lane and the necessity of being in the correct lane in order to achieve specified checkpoints.

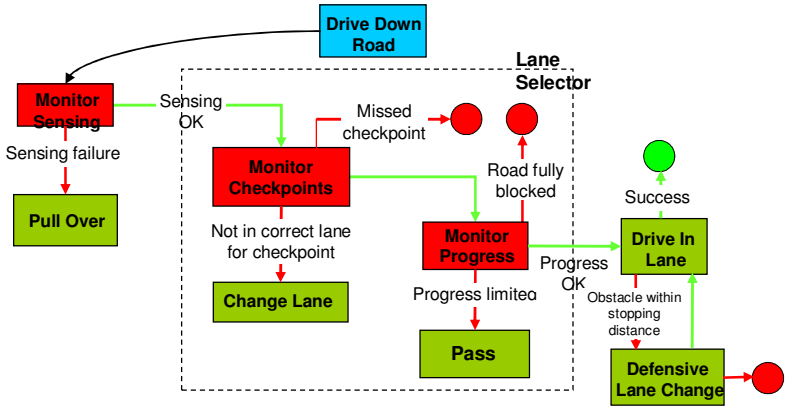


Figure 3. The Drive Down Road behavior. The Lane Selector portion is enclosed by dotted lines.

The distance-keeping behavior aims simultaneously to zero both the difference between our vehicle's velocity and that of the lead vehicle, and the difference between the desired and actual inter-vehicle gaps. The commanded velocity is therefore:

$$v_{cmd} = v_{lead} + K \cdot (gap_{actual} - gap_{desired}) \quad (1)$$

where v_{lead} is the lead-vehicle velocity and K is a settable gain. The desired gap is:

$$gap_{desired} = \max\{(l_{vehicle}/10) \cdot v_{actual}, absMinGap\} \quad (2)$$

where $l_{vehicle}$ is the length of a vehicle, $(l_{vehicle}/10)$ represents the one-vehicle-length-per-10-mph minimum-gapping requirement, and *absMinGap* is the absolute minimum gap requirement. For safety and smoothness, the commanded vehicle acceleration is made proportional to the

difference between the commanded and actual velocities and capped at maximum and minimum values. The distance-keeping behavior has been tested extensively and successfully in simulation and on the vehicle.

The Lane Selector portion of *Drive Down Road* currently has a simple implementation which causes the vehicle to enter the checkpoint lane at a settable distance from the checkpoint. A more comprehensive design that determines feasibility of merging and passing based on the current and predicted state (position, velocity, acceleration) of local traffic and distance to the next checkpoint is nearly complete and will soon be tested in the simulator and on the vehicle. The accompanying logic includes consideration of the Urban Challenge rules for valid vehicle spacing, the current speed limit, and maximum and minimum vehicle acceleration, and specifies a value function embodying lane preferences.

The *Handle Intersection* behavior (Figure 4) establishes a polygonal zone around an intersection and tracks all vehicles within that zone. Those that have arrived at a stop line before our vehicle are yielded precedence. Once our vehicle determines that it has precedence, it checks for the intersection to be clear of obstacles and vehicles completing their exit from the intersection or disobeying precedence rules. It then traverses the intersection in a "virtual lane" created by connecting the exit point of the current lane and the entry point of the goal lane. This portion of the *Handle Intersection* behavior (outlined by a dotted line in the figure) has been tested extensively and successfully in simulation and on the vehicle. Figure 5 shows the *Handle Intersection* behavior in action, operating on the robot using actual perception data. Current work on this behavior involves handling anomalous cases, including near-simultaneous arrival, various types of rule-breaking by other vehicles, and intersection blockage.

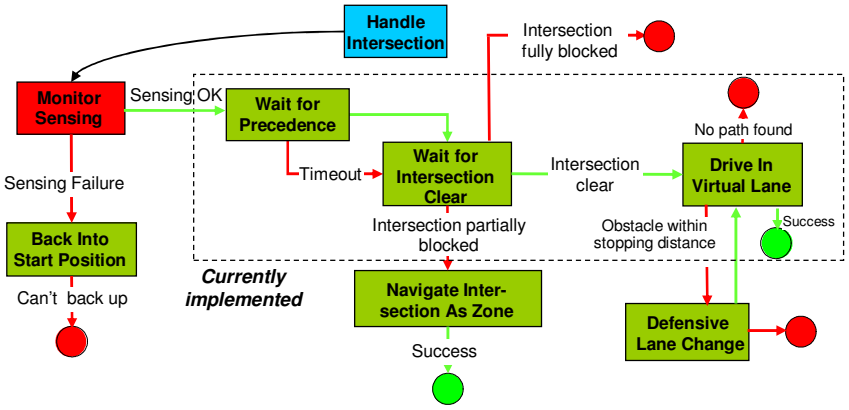


Figure 4. Schematic representation of the "Handle Intersection" behavior.

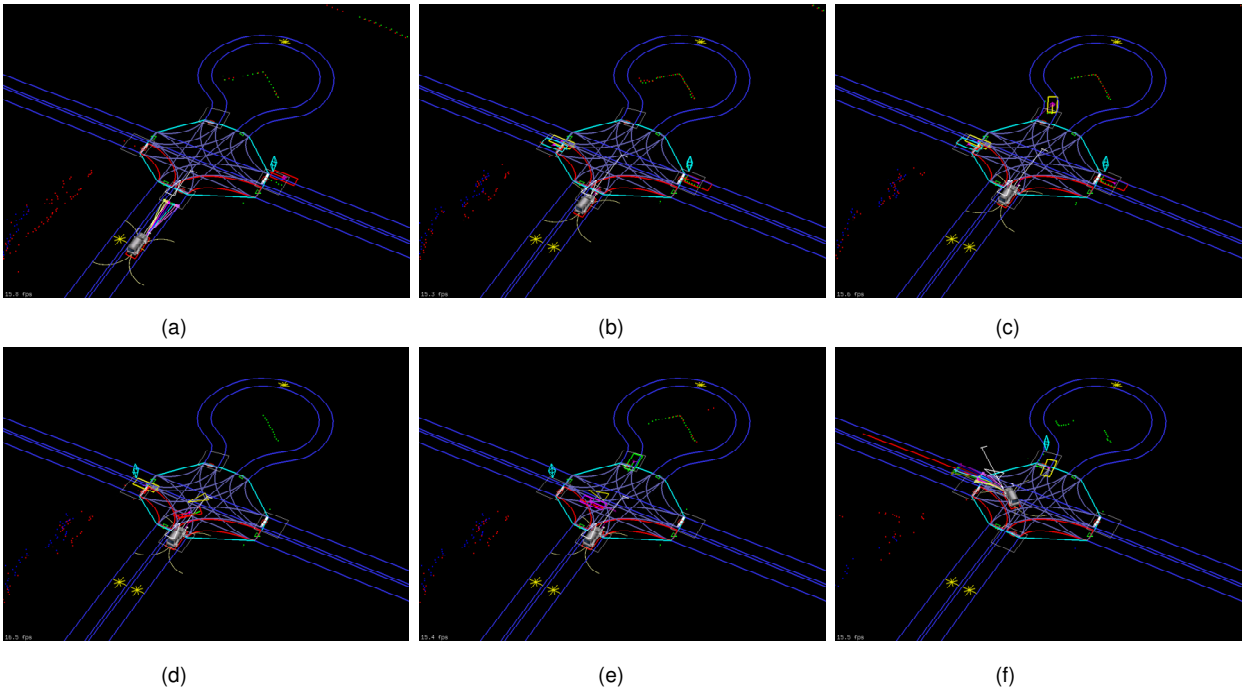


Figure 5. (a) The robot approaches an intersection and identifies a waiting car (rectangle marked by a teal diamond). As the gets closer (b) it detects a second vehicle (rectangle, left of frame), and upon coming to a stop (c) tracks a third vehicle which arrives late (rectangle middle of frame). (d) The first vehicle begins moving through the intersection, and precedence is transferred to the second vehicle, denoted by the teal diamond. (e) The second vehicle begins traversing the intersection. (f) Upon all clear, the robot takes precedence ahead of the third vehicle and safely navigates the intersection.

The *Achieve Zone Pose* behavior specifies a desired pose for the Motion Planner to achieve. In the general case, this behavior is invoked when the system needs to traverse a zone and park the vehicle. As the system matures, this behavior will also be invoked when it is necessary to creatively find its way out of trouble, such as traversing a jammed intersection, or recovering from an off-road position to a position where the *Drive Down Road* behavior can resume.

Motion Planning

In contrast to the strategic level of planning performed by the Mission Planner, the motion planning layer is responsible for executing segments of a route. This typically involves either driving down a lane when on roads or navigating through obstacle fields to a desired goal pose when in zones.

Planning in Lanes

Driving down road lanes relies on the perception sub-system to provide an indication of the current lane boundaries. From this information a curve representing the centerline of the current lane is computed. This is the nominal path the vehicle should follow.

To robustly follow the current lane and to avoid static and dynamic obstacles, a local motion planner generates trajectories to a set of local goals. These goals vary in longitudinal distance from the vehicle and lateral distance from the centerline of the lane to allow for flexible vehicle movement. A trajectory generation algorithm developed by Howard and Kelly [Howard 2006] is

used to compute dynamically feasible trajectories to these local goals. This algorithm uses forward simulation and a high-fidelity vehicle model to accurately predict where the vehicle will end up after following some specified control trajectory. It then modifies the control trajectory in a series of optimization steps to minimize the error between the forwards-simulated vehicle position and the current desired goal position. The resulting optimized trajectories are then evaluated against both static and dynamic obstacles in the environment, as well as their distance from the centerline path, their smoothness, and various other metrics. The best trajectory according to these metrics is selected and executed by the vehicle. Because the trajectory generator computes the feasibility of each trajectory using an accurate vehicle model, the selected trajectory can be directly executed by a vehicle controller.

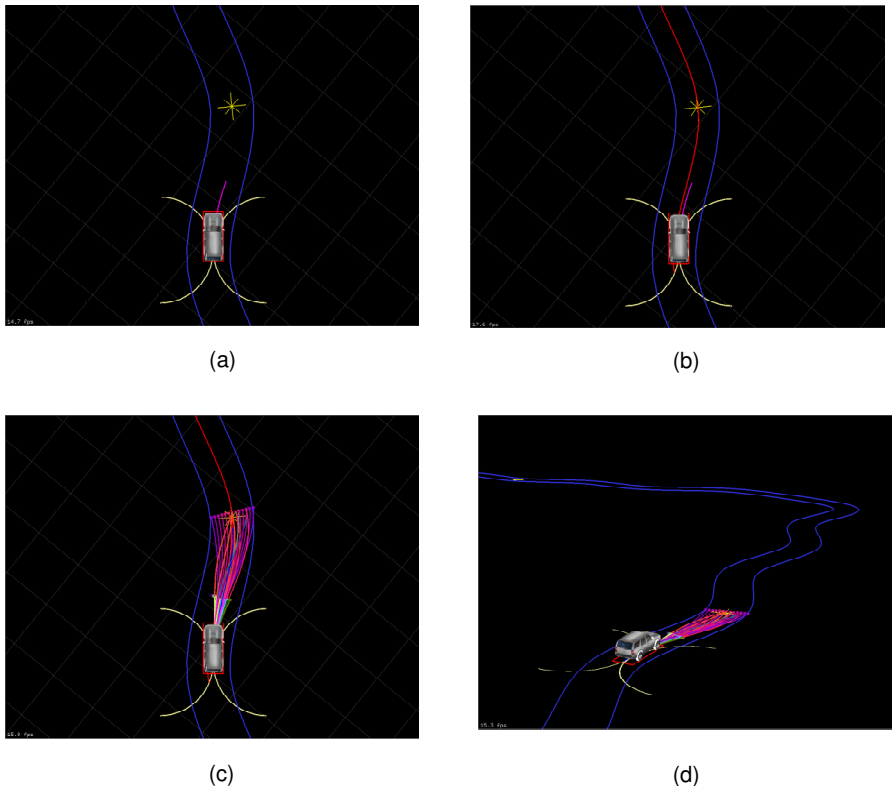


Figure 6. An illustration of the local planner following a road lane.

Figure 6 provides an example of the local planner following a road lane. Figure 6a shows the vehicle navigating down a single-lane road (lane boundaries shown in blue, current curvature of the vehicle shown in pink, minimum turning radius arcs shown in yellow). Figure 6b shows the extracted centerline of the lane (in red). Figure 6c shows candidate trajectories generated by the vehicle given its current state, the centerline path and lane boundaries. A single trajectory is selected for execution, as discussed above. Figure 6d shows the same scenario from an alternative viewing angle.

Planning in Zones

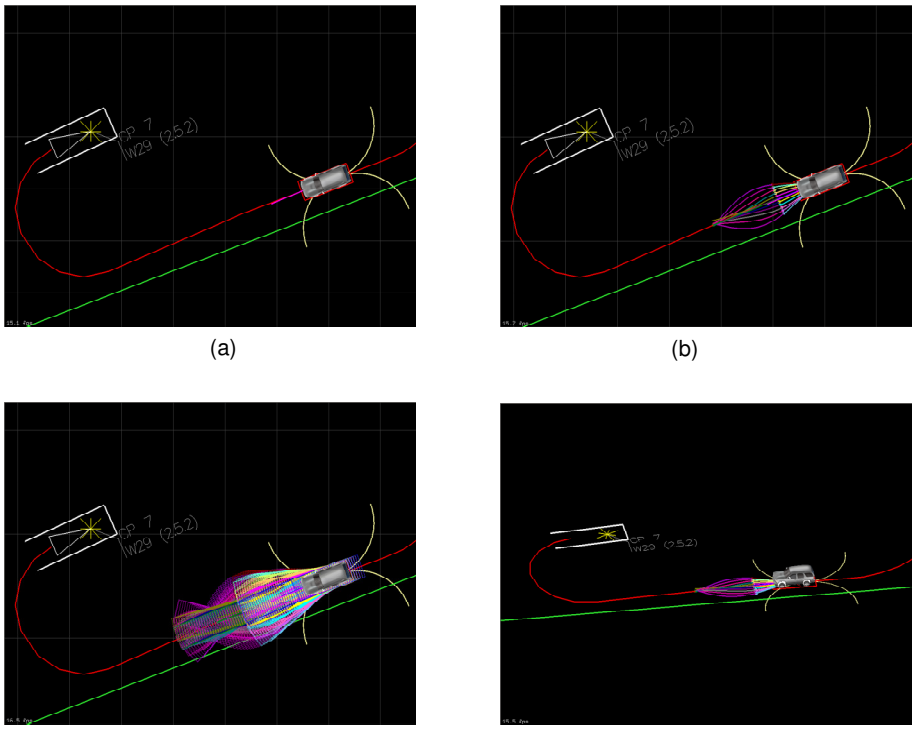
Driving in unstructured environments, such as zones, significantly differs from driving on roads. When traveling on roads, the road lane implicitly provides a preferred pose of the vehicle (typically the centerline of the current lane). In parking lots there are no driving lanes and thus

the movement of the vehicle is far less constrained. Further, it is often the case that in zones we are trying to reach very specific goal poses (such as a particular parking spot) rather than just following a specific lane.

To efficiently plan a path to a distant goal pose in a zone, we use a lattice planner that searches over vehicle position and orientation. The set of possible local maneuvers considered for each state in the planner's search space are constructed offline using the trajectory generator, so they can be accurately executed by the vehicle. This planner searches in a backwards direction, from the goal pose towards the vehicle pose, and generates feasible high-fidelity maneuvers that are collision-free with respect to the static obstacles observed in the environment.

To efficiently generate complex plans over large, obstacle-laden environments, the planner relies on an anytime, replanning search algorithm known as Anytime D*, developed by Likhachev et al. [Likhachev2005]. Anytime D* quickly generates an initial, suboptimal plan for the vehicle and then improves the quality of this solution while deliberation time allows. When new information concerning the obstacles in the environment is received, Anytime D* is able to efficiently repair its existing solution to account for the new obstacle information. This repair process is facilitized by performing the search in a backwards direction, as updated obstacle information in the vicinity of the vehicle therefore affects a smaller portion of the solution. Performing the search in a backwards direction also results in a nearly computation-free re-planning when the vehicle deviates from its path due to tracking errors.

The resulting plan is tracked by the local planner in a similar manner to the paths extracted from road lanes. The local planner generates a set of trajectories that attempt to follow the plan while also allowing for limited deviation to account for dynamic obstacles and execution inaccuracy.



(a) (b)

(c) (d)

Figure 7. An illustration of the local planner following a lattice plan through a zone.

Figure 7 provides an example of the local planner following a lattice plan to a specified parking spot. Figure 7a shows the lattice plan generated for the vehicle (in red) towards the desired parking spot (parking spot boundary shown as white lines, desired pose of the vehicle shown as the white arrow). Figure 7b shows the set of trajectories generated by the vehicle to track this plan, and Figure 7c shows the vehicle bounding boxes overlaid onto the points in these trajectories for collision-checking and cost evaluation by the local planner (this step is also performed during on-road driving). From these trajectories, a single one is selected to be executed by the vehicle. Figure 7d shows the same scenario from an alternative viewing angle.

This lattice planner is flexible enough to be used in a large variety of cases that can occur during the Urban Challenge. For instance, it can be used when navigating congested intersections, to perform U-turns, and to get the vehicle back on track after emergency defensive driving maneuvers.

Perception & World Modeling

Tartan Racing's perception system is responsible for addressing three critical functions. These are: the detection and tracking of moving obstacles, the detection of static obstacles, estimating the shape of the road.

Moving Obstacle Fusion

Moving obstacles are tracked using several lidar and radar. Each of these sensors has different sensing characteristics. Some of the sensors generate raw data while other perform their own internal modeling, prediction, and tracking. The primary job of the sensor fusion module is to provide an abstract way to fuse these different sensor and tracking modalities into a single unified list of tracked objects.

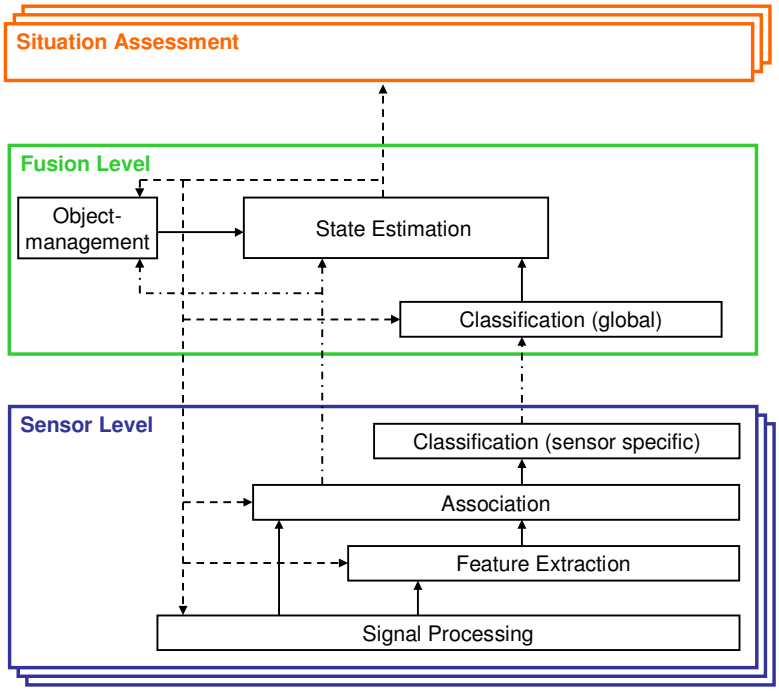


Figure 8. The moving obstacle fusion architecture supports multiple types of sensors.

Similar to an architecture described in [Darms 2005], our moving obstacle fusion architecture consists of three specific layers: sensor, fusion, and situation assessment (see Figure 8).The sensor layer maintains a list of individual sensors whose data will be fused into a single coherent list of tracked objects. Each sensor's data is treated independently, regardless of whether it has the same type (e.g. a SICK lidar) as another. Thus the fusion algorithm is responsible for processing the data from many different input sources. Each measurement is associated with an existing tracked object. Since measurements can be ambiguous, such as in the case of a lidar detecting a small part of the side of a vehicle, there are often multiple ways that the measurement can be associated. Each of these possibilities is analyzed and a quality measurement is generated for each whereby the best global combination of all measurements is chosen. Measurements that do not associate with existing objects will propose the creation of a new object. The sensor layer is also responsible for providing an initial classification of the object which is dependent on the sensor's characteristics. For instance, radars are unable to discern the shape of an object whereas lidar can detect whether the object is shaped like a car. The specific classification of the object also determines the basic dynamic models used by the trackers.

The fusion layer takes each sensor's list of associated (and unassociated) measurements and applies each to the global list of tracked objects. The fusion layer analyzes the proposed object types and decides whether a change to a more explicit object type, such as a vehicle, in to a less-explicit object type such as a more general polygon is warranted based on the available information. If a measurement suggests the creation of a new tracked object that did not exist before, information from the road world model, such as whether the new tracked object is actually near a road, is used as a gating mechanism to avoid the creation of spurious false objects.

At the top level of the architecture is the situation assessment layer. This layer attempts to estimate the "intention" of the tracked object by integrating the estimates with knowledge about the road world model. The planning and behavior modules of the Tartan Racing robots require a mechanism for predicting the potential positions of other vehicles 10 seconds into the future. The individual model-based trackers (described below) are propagated and updated independently of each other and thus are insufficient to handle such long-term prediction, particularly in the case of potential interactions between vehicles at intersections and in parking lots. The situation assessment model depends heavily on the domain in which the tracking is being performed. For instance, intersections will require different top-level semantics from parking lots.

The moving obstacle fusion module has been tested in a wide variety of different situations including (but not limited to) tracking vehicles: in the same lanes, in opposing lanes, performing a 3-point turn in the middle of the road (see Figure 9), driving in very erratic fashions, when our robot is being driven in an erratic fashion, and through traffic circles. This testing has identified several cases where the moving obstacle fusion module generates errors that include improperly tracked objects or initializes false objects. One such problem occurs due to pitch and roll of the vehicle; sensors can either temporarily lose track of an object or inadvertently see the ground plane during a hard braking or acceleration maneuver as well as during sharp turns. This problem will be addressed through filtering based on the vehicles inertial state. A second challenge is correctly classifying objects. The sensor fusion module currently models all returns from objects on the road as vehicles. This problem has occasionally created "false" vehicles that extend out into the roadway and causing the robot to believe that a lane is blocked. The fuser

will be extended to estimate object size to address this problem. Additionally, the fuser will consider the object as only a point object until it is observed to move. At that point, the object will be "upgraded" to a vehicle and tracked as such given that the only moving obstacles relevant to the urban challenge are vehicles.

Moving Obstacle Tracking

The underlying model of the obstacle fusion architecture is a simplified bicycle model that is used to represent the motion of a general vehicle. The dynamic state model is $X = (x, y, \psi, v, \omega, a)$ where (x, y) is the position of the vehicle's center, ψ is the yaw angle, v is the velocity, ω is the yaw rate, and a is the linear acceleration. An Extended Kalman Filter [Bar-Shalom2001] state estimator is used to predict and update the state and uncertainty measurement for each tracked object. Our approach makes use of a square root covariance scheme which is more numerically stable than a traditional EKF formulation due to the use of the so-called UdU factorization of the covariance matrix [Bierman1977]. The propagation model used by the architecture approximates constant acceleration and constant yaw rate.

The individual sensor modalities each use specific algorithms to generate meaningful object observations from the raw data. For instance, the three bumper-mounted SICK lidar use an algorithm similar to [Mertz2005] to cluster raw lidar returns into linear segments (single edge) and convex 90 degree corner (double edge) objects.

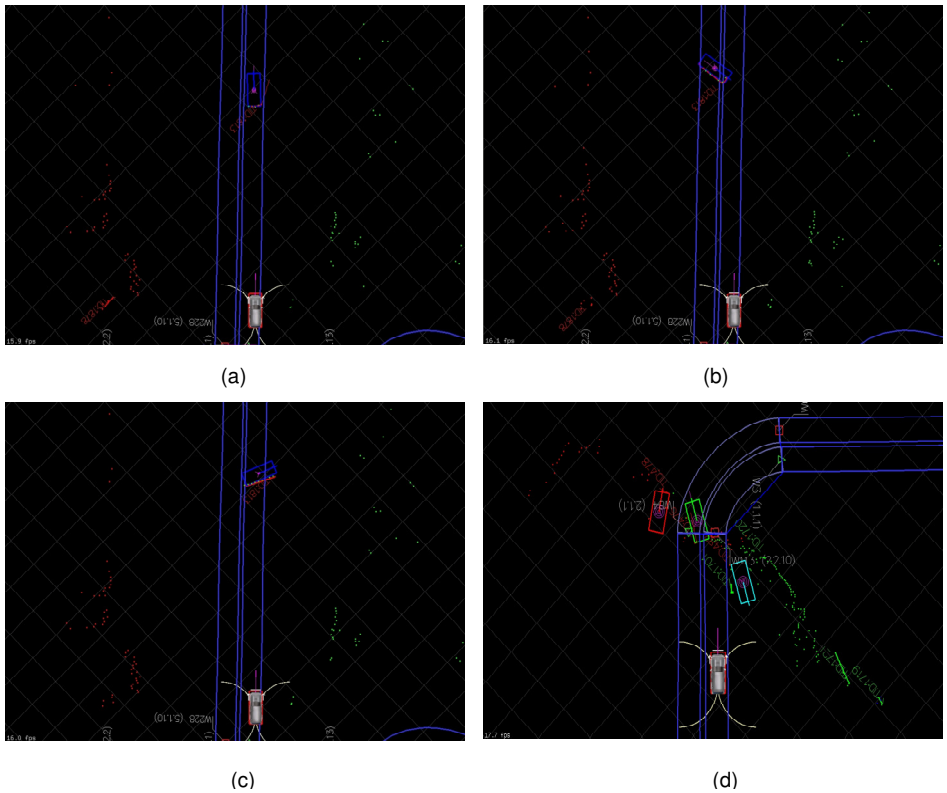


Figure 9. The moving obstacle fusion module following and tracking a vehicle as it performs a U-turn (a)-(c). The tracked vehicle is shown as a blue rectangle. In (d), a hard brake maneuver pitches the robot's sensors down causing "ghost" objects to appear due to sensor reflections from the ground.

Static Obstacle Detection

The static obstacle detection algorithm uses downward looking laser sensors mounted on the roof of the robot to evaluate the terrain around the vehicle. The algorithm computes a cost map representing the "traversability" of the terrain. The core of the algorithm relies on comparisons of pairs of laser points. This comparison calculates a cost based on the elevation difference between the two points and the angle of the vector connecting them, relative to 'ground'. The maximum elevation and angle differences are tunable parameters adjusted such that a cost of '1' is considered 'fatal'. Each incoming laser point is compared to all of its neighbors that contain points and the maximum cost is recorded as the cost of that point. The maximum cost for all points in a cell is maintained and placed in a cost map.

To ensure that obstacles of a certain height are seen, the system maintains a history of points spanning a constant distance traveled by the vehicle, which ensures downward looking lasers to also scan a constant vertical distance for each planar location To prevent storing excess points, laser scans are skipped with increasing frequency as vehicle speed drops. The cost map is updated only in areas where a new point is inserted as costs would only change in these areas. So that terrain costs are not forgotten if a new laser point is inserted when all neighbor points have expired, the maximum cost for a cell is maintained for much longer than individual point. When a moving obstacle is detected the cost map is cleared of obstacles in that area. As the cost map is only updated where new points are inserted, this is sufficient for removing the 'snake' of obstacles along the object's path.

The static obstacle detection algorithm combined with the current sensor calibration has proven sufficient for navigating obstacle courses of walls and cones with minimal errors. Figure 10 shows a picture of a scene with a building, fence, and various objects along with an image of the algorithm's interpretation of the scene.

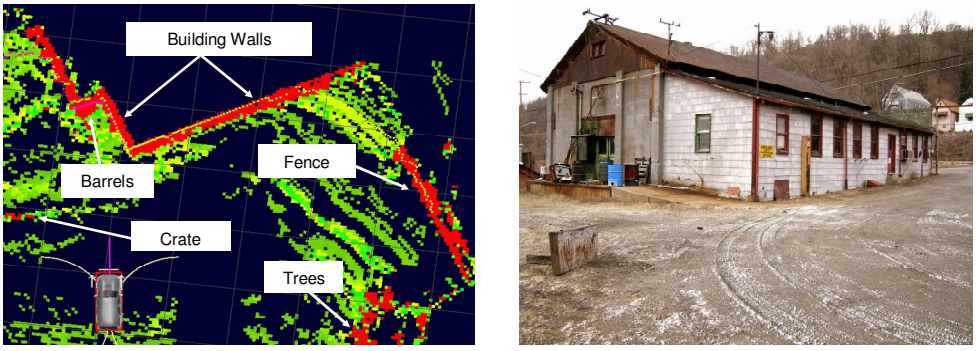


Figure 10. An illustration of the static obstacle map generation and its limitations.

In this coloring scheme, green is safe terrain, yellow is less desirable and red is 'lethal'. The walls of the building and fence (out of the picture to the right) are clearly visible in the cost map as solid lines of red. Additionally, the trees, crate and barrels also appear as lethal obstacles.

The above figure was chosen to illustrate some flaws in the algorithm, specifically its weakness in the face of poor calibration. The significant thickness of the walls in the cost map and the yellow (undesirable) pixels between the building and fence where the road is still moderately flat are due to improperly transforming laser data to world space. The algorithm also sees foliage as obstacles, note the 'lethal' areas by the ramp caused by grass on the ground, a trait exacerbated

by poor calibration. An additional weakness of the algorithm is its susceptibility to missing hanging cables.

The accuracy of obstacle placement is clearly dependent on the sensor calibration and the resolution of the cost map. At present, the algorithm properly places cones within 25 cm of their location. In most cases this is sufficient to navigate through obstacle fields with cones and walls.

Road Shape Estimation

Urban driving is distinguished from off-road driving by the need to understand and respect lane boundaries. In the context of the Urban Challenge, a strong prior model for the road shape is provided for much of the environment, via a route network description file and aerial imagery. However, due to potentially outdated aerial imagery and the absolute accuracy GPS, road lanes cannot be sufficiently located without perception. To estimate road shape, both offline and real time perception techniques are needed. Off-line, a learning based approach is used to combine the route network information with aerial imagery to improve prior road shape estimates. The algorithm is an extension of the Maximum Margin Planning and structured boosting approaches of [Ratliff2006, Ratliff2007]. This approach builds planning algorithms directly into the image processing: it allows planning in the image space between waypoints to extract road segments. The current algorithm combines image filters, convolution and neural network classifiers, as well as an efficient planning algorithm. A key to our approach is that improving any component should help our ability to extract roads: better filters make learning easier, better training algorithms improve generalization and performance, and better planning takes into account the types of maneuvers that real vehicles make on real roads. Figure 11 shows some example roads, and the extracted road model.

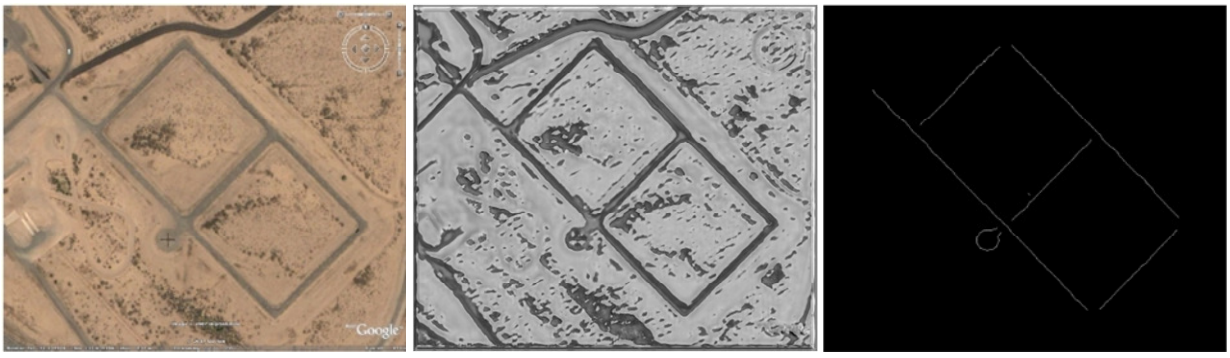


Figure 11. Illustrates the off-line road shape extraction algorithm, (a) is an image of a road network, (b) shows the learned cost function and (c) shows the extracted road network.

Road Shape Feature Detectors

Online Road shape detection is primarily generated using range and intensity data from down-looking short range lidars. Changes in the intensity of the returned laser data is indicative of painted road lines. In addition, relative changes from flat surfaces to raised curbs or drops to soft shoulders can be detected by looking for the appropriate geometric features. These two cues are used to detect the boundaries of the lanes and thus the overall shape of the road. Several algorithms identify the shape of the road from the returned lidar data. These include Haar wavelet transforms, heuristic edge detection with adaptive thresholding, and dynamic

programming methods for optimal line splitting. In the interest of space, only the first of these will be described here.

Wavelet transforms provide a technique for efficiently performing multi-resolution analysis of a signal [Daubechies1992]. A base template, known as a mother wavelet, is chosen for a given transform; in this case, the Haar wavelet. The mother wavelet is used to generate a series of daughter wavelets, which are stretched versions of the mother wavelet. These daughter wavelets represent the template at different frequencies, and by sliding these daughter wavelets over the input signal and recording a measurement of the similarity between daughter wavelet and input signal, a multi-resolution template matching is performed. The Haar wavelet represents a numerical derivative as a template, resulting in a wavelet transform which performs derivatives at multiple resolutions [Haar 1910].

This approach leverages two key points: at coarse resolutions, the difference between roads and curbs tend to be significantly magnified, and a road tends to be uniform over a short region. The basic algorithm performs a Haar wavelet transform over the heights of points in a lidar scan. The resultant derivatives are then taken at the coarsest resolution, and classified into either road candidates or curb candidates. These candidates are then expanded into the higher resolution derivatives, and a statistical model of the road is built from the road candidates. This model is used to reclassify candidate points, and the process is repeated until the highest resolution is reached.

Examples of the algorithm's performance can be seen in the Figure 12 below, which display examples of output accumulated from this algorithm. In these images, white points represent the detected locations of the curb, with blue and green points representing LIDAR points collected from two vertically mounted, sideways looking SICK lasers. The algorithm clearly detects the road edges, even on banked curves with berms, or when the curb is not easily identifiable. Additional work in improving the filtering of detected points can narrow the range of detected curbs and improve robustness in ill-defined scenarios such as roads with roads surrounded by scattered brush.

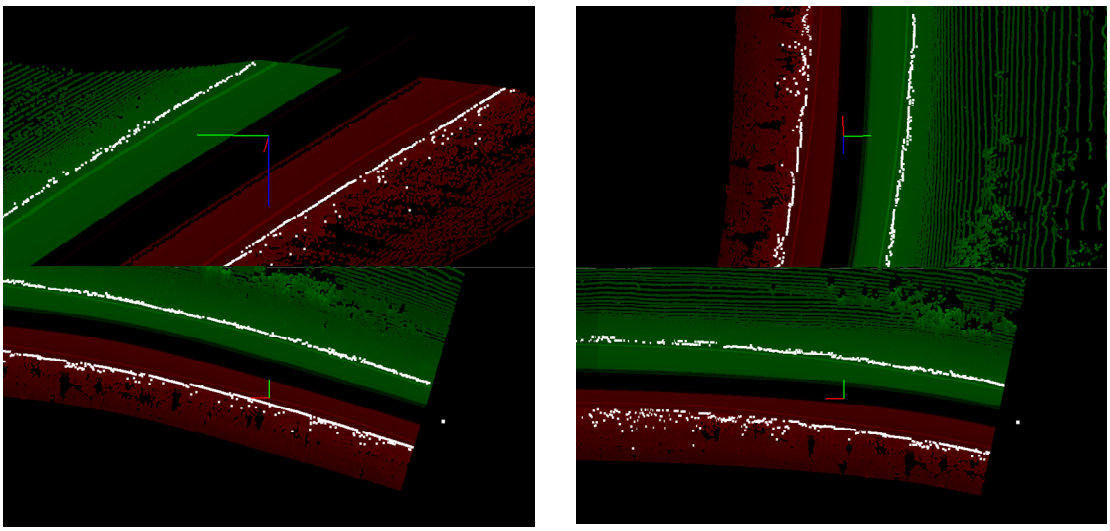


Figure 12. Examples of the Haar road edge detection algorithm operating on stretches of straight and banked roads. Green and red dots show data from driver and passenger side looking SICK lasers. The detected road edge is marked white.

The electrical and mechanical systems that constitute Boss (the Tartan Racing robot) (see Figure 2) use a combination of off-the-shelf and custom developed components, selected with an emphasis on reliability and functionality. The chassis is a Chevrolet Tahoe which was selected due to its integrated electronic interfaces, its high roof giving a good sensor perspective for detecting other vehicles, and its plentiful room on the inside for both developers and additional electronics.

Mechatronic alterations include systems for vehicle automation, auxiliary power, computing, and sensor mounting. Proven technology is used where available, with custom designs used only where necessary to gain significant performance improvements.

Vehicle Automation

Steering and brake/throttle are actuated by closed-loop control of motors acting on the steering column and brake and gas pedals. Low level control of theses motors, along with the actuation of secondary driving controls (turn signals, transmission shifting, etc.) is performed utilizing a commercial-off-the-shelf system from Electronic Mobility Controls (EMC).

While EMC provides physical controls for braking and steering, higher level control is still needed for driving. An additional hard real-time embedded control unit accomplishes this task. This controller is responsible for vehicle functions such as velocity control and curvature control, as well as making sure the vehicle is safe in E-stop conditions.

The real time controller receives vehicle control commands such as desired curvature and speed through a CAN bus and closes the control loop based on the error between commands and actual status with on-board sensors. For curvature control, the control loop is primarily feedfoward control, supplemented by integral feedback for steady-state error correction. The speed control structure includes two non-linear PID controllers one for throttle and one for brake, with logic that switches between the two, depending on the speed error. The switched controller scheme is necessary since:

- throttle and brake performances of a vehicle are dramatically different. The brake system responds much faster to deceleration commands than the throttle system does to acceleration commands (controller/mechatronics or machine as a whole?)
- the actuator for throttle and brake is a single motor system.

Therefore, switching logic is used to move the actuator off the throttle to the brake as soon as possible, and vice versa, when such a need rises. Figure 13 shows commanded and actual vehicle responses in curvature and speed controls.

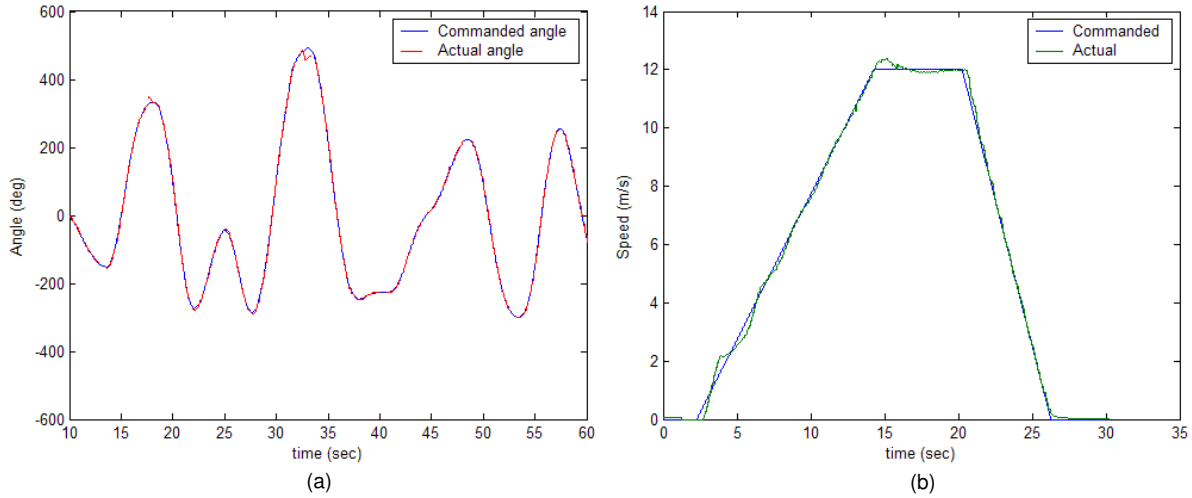


Figure 13.Representative steering angle and speed response curves.

Power Electronics

The Tartan Racing robots utilize auxiliary power generation to support sensors and computing. To avoid interfering with base vehicle functions, and to provide a safe, highly reliable power source for low-level vehicle controls and the safety system, the stock vehicle power bus remains intact. The auxiliary power system is a high voltage generator that is driven by the engine using a secondary serpentine belt.

This generator provides up to 6 kilowatts of power, (which is considerably more than currently needed) depending on engine load. It does this through the use of an Integrated Control System, which contains a series of power converters that convert the power from the high voltage produced by the generator to both direct current (DC) busses and alternating (AC) busses needed by the electronic equipment, while constantly managing power output, voltage levels, and temperatures of the generator and converters

Power is distributed to components through an industrial grade vehicle electrical center. This is a multi-function panel containing fuses, relays, and circuit breakers to protect components and subsystems on the vehicle. Electrical circuits are organized in such a way that a single blown circuit will never completely blind or disable any sensor coverage areas. As an additional precaution, sets of circuits/sensors are sequentially powered up, so that there is not a large voltage drop from powering-up the complete system simultaneously. This also prevents startup errors on data-buses where multiple components are starting up at once.

Sensors

Selecting the right sensors is critical to efficient and safe autonomous urban driving. Sensing horizons and fields of view are driven by the need to detect traffic while crossing an intersection or passing an obstacle in an opposing traffic lane. To understand these requirements a series of experiments were performed to characterize the time required to make left and right turns, merge into 30mph traffic, and to pull out, pass, and re-enter a traffic lane while avoiding a static obstacle. Figures 14 and 15 illustrate the timing and required sensing horizons for making a left turn across traffic and passing a stopped vehicle.

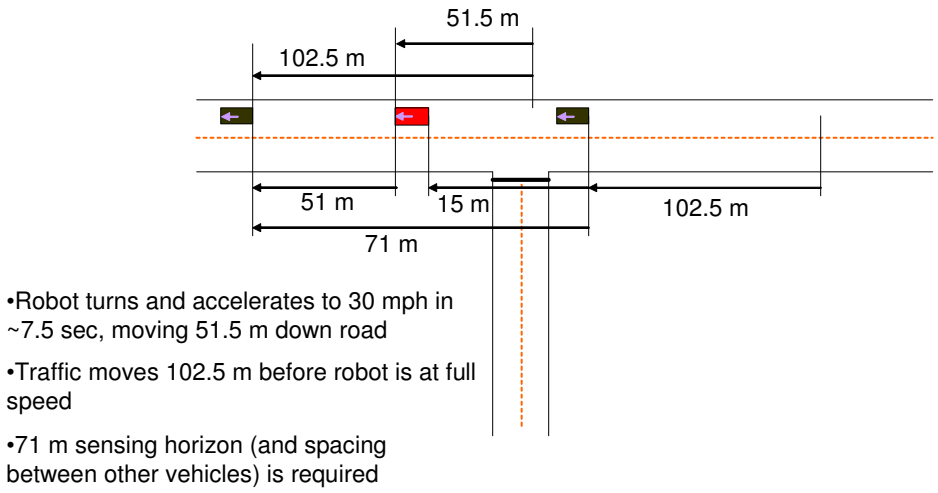


Figure 14. Making a left-hand turn into 30mph traffic requires a 71m sensing horizon for a stock Chevrolet Tahoe

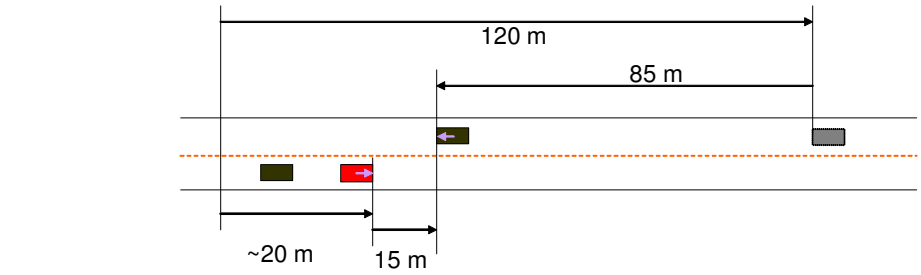


Figure 15. Passing a stopped vehicle with 30mph oncoming traffic requires a 120m sensing horizon for a stock Chevrolet Tahoe.

To meet these requirements in an arbitrary urban environment (where planar and vertical curvature are not constrained by highway codes) requires a 360° planar field of view with a large vertical field of view. The tasks the perception system must perform include:

- Determine it is safe to cross or merge with traffic at an intersection (>65m horizon)
- Determine it is safe to pass a static obstacle in an oncoming traffic lane (>120m horizon)
- Detect, localize and track vehicles such to hold separation distances (<30m horizon)
- Estimate road shape and lane locations (<30m horizon)
- Provide robust, redundant and complimentary sensing where possible

Required Vertical Field of View

The vertical field of view requirement is driven by the maximum change in grade anticipated on the course. To bound the design, it is assumed that the maximum change in grade will be limited to 5% over an arbitrary distance greater than our maximum sensing horizon. A 5% change in

grade implies a required vertical sensor field of view of +/-2.8°.

Required Lidar Horizontal Angular Resolution

To detect an object reliably it is necessary to have at least two lidar returns from that object. Given this requirement, and the assumption that vehicles are nominally 2m wide, the required horizontal angular resolution for a lidar sensor can be expressed as a function of range as:

$$\vartheta_h = \frac{1}{R} \tag{3}$$

Given the range requirements for passing an obstacle in lane, and crossing an intersection, it is necessary to incorporate sensors with an angular resolution of better than 0.5°.

Scan Frequency

To track an object data must be associated between frames. There is often some upper bound on the distance between the image of an object in sequential frames over which data association can be performed efficiently. Since no vehicles are allowed to move faster than 13.4m/s and we are able to correct for the motion of our own vehicle, the required scan frequency from a sensor given a maximum allowable inter-frame motion in meters (s_{max}):

$$f = 13.4 / s_{\max} \tag{4}$$

Therefore, if we were to allow 1m of motion between frames, we would require a 13.4hz sensor.

Sensors

The Tartan Racing vehicles use a combination of sensors to both the necessary sensing requirements and some redundancy should sensors fail during operation. The selected sensors are described in Table 1, and their role in the overall perception strategy is outlined in Table 2.

Table 1. A description of the sensors incorporated onto the Tartan Racing Robots.

Sensor	Characteristics
Applanix POS-LV 220/420 GPS / IMU	• sub-meter accuracy with Omnistar VBS corrections • tightly coupled inertial/GPS bridges GPS-outages
SICK LMS 291-S05/S14 Lidar	• 180º / 90º x 0.9º FOV with 1º / 0.5º angular resolution • 80m maximum range
Velodyne HDL-64 Lidar	• 360º x 26º FOV with 0.1º angular resolution • 70m maximum range
Continental ISF 172 Lidar	• 12º x 3.2º FOV • 150m maximum range
IBEO Alasca XT Lidar	• 240º x 3.2º FOV • 300m maximum range
Ma/Com Radar	• 80º FOV • 27m maximum range
Continental ARS 300 Radar	• 60º / 17º x 3.2º FOV • 60m / 200m maximum range
MobilEye Vision System	• 45º x 45º FOV • ~35m effective range

Table 2. The roles filled by different sensing modalities. Green indicates that the sensor is fully capable of filling this role, yellow indicates the sensor is capable given speed limitations.

Role	Velodyne	SICK	Alaska XT	ISF 172	ARS 300	Ma/COM Radar	Mobileye	Camera
Determine safe to cross/merge at intersection								
Determine safe to pass in oncoming traffic								
Detection & localize vehicles for separation								
Estimate road shape and lane locations								
Detection of static obstacles in the road								

Integration and Testing

A comprehensive test and evaluation strategy is the pathway to a robust and reliable autonomous vehicle. This strategy is as important to safe urban driving as the algorithmic advancements. Our process involves incrementally testing each component, as well as the system as a whole, as functionality is added and capabilities mature. The software is tested at a unit, subsystem, and system level using a number of standard scenarios. Each scenario specifies a particular RNDF and MDF that the system must execute along with various traffic and obstacle configurations.

Simulation capabilities include the specification of scenarios that simulate traffic in order to develop and test the system off the vehicle. Additionally, prior to testing with live traffic, the system can be configured to operate on a vehicle with simulation-in-the-loop, which provides a safe testing mechanism, prior to testing with real obstacles and traffic.

The system is continuously subjected to five tiers of testing, ranging from the component level to full fledged on-vehicle testing with live traffic:

Unit Testing – Unit tests are small, simple software tests written by the developer while writing a particular class or function. Each individual test exercises a class method or function using a set of "interesting" inputs. The objective is to verify that the class method or function being tested produces the correct state changes or output for the provided set of inputs.

Subsystem Testing with Simulated Inputs – The software infrastructure includes functionality that allows any developer to easily log data from any software interface. This data can later be played back for analysis, testing, debugging, etc. A particular subsystem or individual process (task) can be run in isolation from the rest of the system while its interfaces are "simulated" by playing back data. For example, a new static obstacle detection algorithm is tested easily by running the process in isolation, and playing back vehicle state (position information) and raw lidar data. The obstacle detection algorithm receives the played back data through its interface as if it were talking to the real sensors, and the cost map is displayed in a debug tool for analysis. Testing the component in isolation made for a much easier integration with the rest of the system by finding bugs earlier in the process.

System Testing in Pure Simulation – The entire system can be executed on multiple computers while the various on-vehicle interfaces are simulated. These interfaces can be simulated by playing back previously logged data such as lidar data, cost maps, vehicle state, or any other interface. Additionally, a simulated robot can be configured to respond to vehicle commands, close the loop on position, and sending back an appropriate vehicle state message to the system. Any number of other simulated robots can also be run in the system according to a particular scripted behavior, in order to simulate live traffic (moving objects).

System Testing on Vehicle with Simulation In the Loop – The system is executed on the robot, but rather than using live traffic and real perception to sense/track that traffic, simulation is used on board the robot to present simulated moving objects to the real system. This allows the robot to 'see' and react to this virtual traffic without the risk of testing in the presence of other vehicles.

System Testing on Vehicle with Live Traffic – The full-fledged system is executed in the presence of real traffic.

Table 3. TEC Elements have been divided among 8 system tests.

Test #	Description	TEC Elements
1	Checkpoints & Safety	**A1** Preparation for Run **A2** Mission Start **A3** Checkpoints Warning Devices etc.
2	Intersection Handling	**A4** Stay in Lane **A6** Excess Delay **A8** Stop Lines **B2** Intersection Precedence
3	Separation in Lanes, Passing & Speed Limits	**A5** Speed Limits **A9** Vehicle Separation
4	Obstacle Avoidance, Parking & U-turns	**A10** Leaving Lane to Pass **A11** Returning to Lane After Pass **B3** Minimum Following Distance **B4** Queuing
5	GPS loss and Road Following	**C5** Road Following **C6** GPS Outages
6	Passing with Moving Obstacles	**D6** Zones **D7** Emergency Braking
7	Merging & Crossing Moving Traffic	**D2** Merge **D3** Vehicle Separation During Merge **D4** Left Turn **D5** Vehicle Separation During Left Turn
8	Route Blockage & Replanning	**C4** Dynamic Re-planning **D8** Defensive Driving **D9** Traffic Jam

As part of the test process, standardized tests are completed once a week to evaluate system performance. Tests are designed to judge capability relative to the Urban Challenge technical evaluation criteria and internally derived requirements. Eight tests span the capabilities required for the Urban Challenge, as outlined in Table 3. Frequent testing provides an up-to-the-minute

understanding of the capability of the overall system.

To date, six weekly tests have been performed, progressing through the first four system tests. Each test is developed by a team external to the core development group. Tests are designed to critically evaluate performance and challenge capability. As an example, Figure 16 illustrates the obstacle configuration for the Obstacle Avoidance and Parking lot tests.

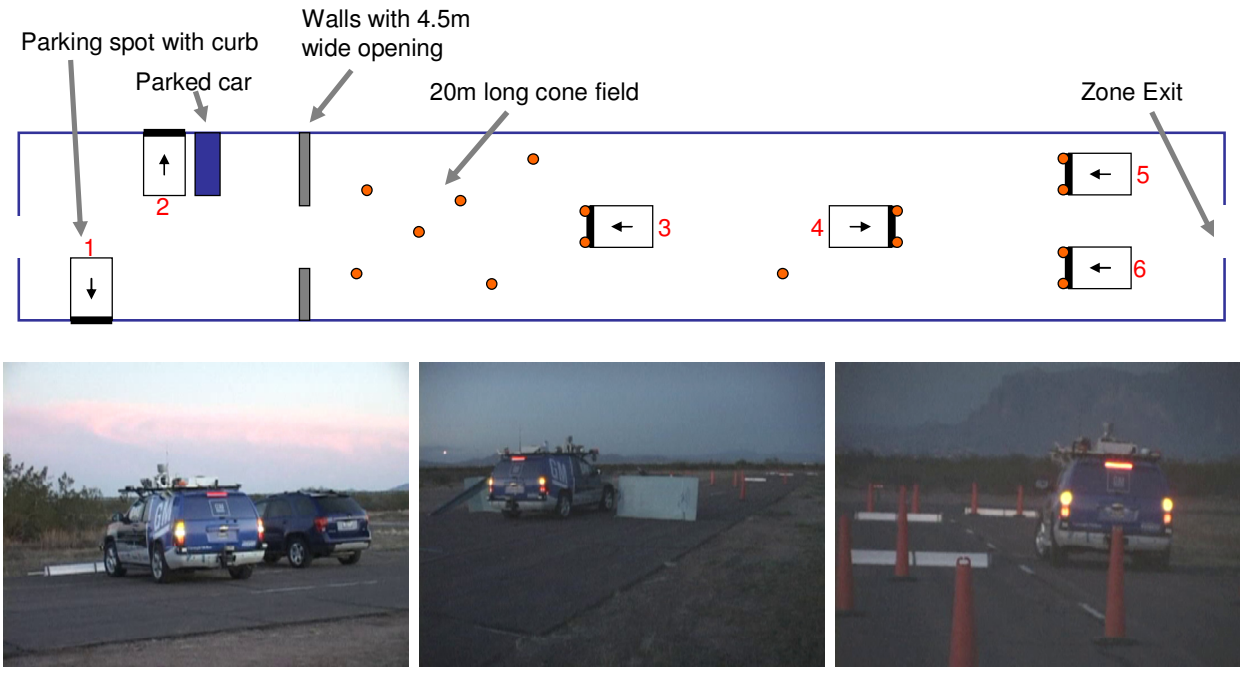


Figure 16. The standard parking lot test includes a challenging obstacle field and a narrow opening. During system tests, the robot has successfully navigated this parking lot.

These tests represent a superset of the capabilities required for the Urban Challenge site visit. Figure 17 shows the growth of capability. As an example of the benefit of rigorous system testing consider the dip in performance during system test six. The moving obstacle fusion algorithm and an updated velocity controller were integrated for the first time. Despite rigorous component testing, the introduction of these elements reduced overall system capability. Through regression testing problems were identified and addressed quickly, and it is expected that the next system test will show a return to the previous level of performance. As development continues and the overall system complexity increases, standardized regression tests at the system level will play an even larger role in identifying problems with components as they are integrated.

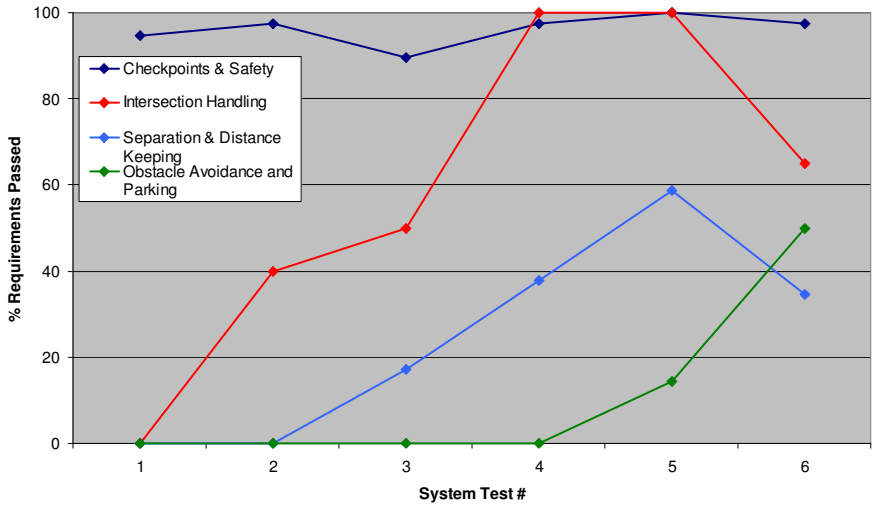


Figure 17. Standardized system tests provide a critical monitor of the overall development health and allow for the rapid diagnosis of problems with newly introduced components.

Summary

This paper describes the approach Tartan Racing is taking to meeting the Urban Challenge and realizing the goal of safe, autonomous vehicles. Our multi-modal approach partitions the urban driving problem into tractable behavioral modes (road driving, intersection handling and parking). Motion planning employs a common trajectory generation framework for both on road and parking lot planning. Perception integrates data from radar, lidar and vision to track vehicles and model road shape and obstacle locations.

The results presented is a snapshot of technology under rapid development. Significant improvements are ongoing in all aspects of overall system performance. More research and development is occurring that could not be presented in this paper in the interest of clarity and space. In particular several perception algorithms, which are still early in the development cycle, have not been presented.

As of this publication, the Tartan Racing vehicles have exhibited a majority of the basic navigation skills required for the site visit, and the building blocks necessary to achieve the more challenging navigational components are assembled. We anticipate a tremendously exciting run at the Urban Challenge.

References

Y. Bar-Shalom, “Estimation with Applications to Tracking and Navigation”, Wiley-Interscience, 2001.

G. Bierman, “Factorization Methods for Discrete Sequential Estimation”, Academic Press, 1977.

M. Darms, & H. Winner, “A Modular System Architecture for Sensor Data Processing of ADAS Applications”. In Intelligent Vehicles Symposium, 2005. Proceedings. IEEE. Las Vegas, USA, 6.-8.6.2005 2005, 729 – 734.

I. Daubechies, *Ten Lectures On Wavelets*. 2nd ed. Philadelphia: SIAM, 1992.

A. Haar, *Zur Theorie der orthogonalen Funktionensysteme*, Mathematische Annalen, **69**, pp 331-371, 1910.

T. Howard, R. Knepper, and A. Kelly, "Constrained Optimization Path Following of Wheeled Robots in Natural Terrain," ISER 2006: The Tenth International Symposium on Experimental Robotics, 2006.

M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun, “Anytime Dynamic A*: An Anytime, Replanning Algorithm,” International Conference on Automated Planning and Scheduling (ICAPS), 2005.

C. Mertz, D. Duggins, J. Gowdy, J. Kozar, R. MacLachlan, A.M. Steinfeld, A. Suppe, C. Thorpe, and C. Wang, “Collision Warning and Sensor Data Processing in Urban Areas,” In Proceedings of the 5th international conference on ITS telecommunications, June, 2005, pp. 73-78.

N. Ratliff, D. Bradley, J. Bagnell, and J. Chestnutt, “Boosting Structured Prediction for Imitation Learning”, in Advances in Neural Information Processing Systems 19, MIT Press, Cambridge, MA, 2007.

N. Ratliff, J. Bagnell, and M. Zinkevich, “Maximum Margin Planning”, in International Conference on Machine Learning, July, 2006.