Development of Autonomous Car – Part II: A Case Study on the Implementation of an Autonomous Driving System Based on Distributed Architecture

Kichun Jo, *Member, IEEE*, Junsoo Kim, *Student Member, IEEE*, Dongchul Kim, Chulhoon Jang, *Student Member, IEEE* and Myoungho Sunwoo, *Member, IEEE*

***Abstract*— Part I of this paper proposed a development process and system platform for the development of autonomous cars based on distributed system architecture. The proposed development methodology enabled the design and development of an autonomous car with benefits such as a reduction of computational complexity, fault-tolerant characteristics, and system modularity. In present paper (Part II), a case study of the proposed development methodology are addressed by showing the implementation process of an autonomous driving system. In order to describe the implementation process intuitively, core autonomous driving algorithms (localization, perception, planning, vehicle control, and system management) are briefly introduced and applied to the implementation of an autonomous driving system. We are able to examine the advantages of a distributed system architecture and the proposed development process by conducting a case study on the autonomous system implementation. The validity of the proposed methodology is proved through the autonomous car, A1 that won the 2012 Autonomous Vehicle Competition in Korea with all missions completed.**

***Index Terms*—Autonomous car, distributed system, development process, system platform, FlexRay**

I. INTRODUCTION

SAFETY regulations in the automotive industry have become increasingly stringent along with the growing customer demands for comfort functionality. For this reason, the development of intelligent vehicle technology has been accelerated. The ultimate goal of intelligent vehicle technology is the realization of an autonomous car that can drive itself without collisions by perceiving the surrounding environment.

The Defense Advanced Research Projects Agency (DARPA) opened the Grand and Urban Challenge competition in the USA to stimulate research for autonomous cars [1-4]. The competition helped promote the feasibility of autonomous cars; consequently, global automakers and IT companies have endeavored to develop commercial autonomous cars.

In Korea, two Autonomous Vehicle Competitions (AVCs) were held in 2010 and 2012 by Hyundai Motor Group in order to establish the foundation of autonomous driving technology. The 2010 AVC focused on fundamental technology such as waypoint tracking and static obstacle avoidance [5, 6]. After two years, the 2012 AVC promoted the development of autonomous driving technology in urban driving environments containing elements such as traffic signals, moving vehicles, and pedestrians [7, 8]. This paper (Part II of a two-part paper) represents the results of autonomous car A1, which was the winner of the 2012 AVC.

In Part I of this two-part paper [9], we proposed a development process for a distributed system. The development process was reformed to improve flexibility and scalability as well as discard unnecessary processes based on an AUTomotive Open System ARchitecture (AUTOSAR) methodology and platform [10-14]. The process consists of three steps (software component design, computing unit mapping and implementation of functions). In the software component design step, software components for autonomous driving were designed and the flow of information between the components was defined with a Virtual Functional Bus (VFB), which is an abstraction of communication. Second, the designed software components were mapped to each distributed computing unit by considering the computing power, mounting position, and operating systems. Finally, a software platform implemented the software components in the assigned computing unit.

Part II of this paper presents a case study on the implementation of autonomous driving system based on the *development process and distributed system architecture* which were introduced in Part I by using an autonomous car, A1. Various autonomous driving algorithms (perception, localization, planning, and control) and system components of autonomous cars (many heterogeneous sensors, actuators, and computers) are integrated and implemented based on the proposed development process. Each step of the development process (software components design, computing unit mapping, and implementation of functions) is explained using the implementation example of autonomous car A1. The

Manuscript received November 27, 2013; revised April 10, 2014, July 14, 2014, October 24, 2014, October 27, 2014 and January 23, 2015; accepted January 24, 2015.

This work was supported by an NRF grant funded by the Korean government (MEST) (No. 2011-0017495), Industrial Strategy Technology Development Program of MKE (No. 10039673), and BK21 plus program (22A20130000045) under the Ministry of Education, Republic of Korea.

K. Jo, J. Kim, D. Kim and C. Jang are with the Automotive Control and Electronics Laboratory (ACE Lab), Department of Automotive Engineering, Hanyang University, Seoul 133-791, Korea

M. Sunwoo is a director of the ACE Lab, Department of Automotive Engineering, Hanyang University, Seoul 133-791, Korea (corresponding author to provide phone: 82-2-2220-0453; fax: 82-2-2297-5495; e-mail: msunwoo@hanyang.ac.kr).

implementation results show that we can obtain many benefits by using the proposed development process and distributed system architecture.

This paper is organized as follows. Section II presents the 2012 autonomous vehicle competition, and Section III provides a system overview and the autonomous driving algorithm of A1. Section IV presents the implementation of the autonomous driving algorithm with the proposed process and platform. Section V presents the autonomous driving results of A1, and Section VI provides the conclusions. The logical flow and experimental results of Part II are tightly coupled with Part I; therefore, it is recommended to read the development methodology of Part I before referring to this paper.

II. Introduction of 2012 AVC

To enhance the research of autonomous driving systems, the 2012 AVC aimed at the development of autonomous cars for urban driving. The missions are composed of urban driving scenarios such as traffic signal detection and overtaking slow vehicles, as shown in Table I. Therefore, a participating autonomous car should not only drive a race track by itself but also complete the various missions. The race track consists of 1.6 km on-road and 1.8 km off-road, as shown in Fig. 1. The red and blue dots indicate start and finish position, and the arrows in the race track represent the driving directions. Track related map information (including waypoints) were provided by the AVC organizers. During the race most urban-related missions were performed on-road; alternatively, harsh and emergency driving condition were performed off-road.

Penalties or additional credits are defined for each mission as shown in Table I. The purpose of the traffic light and crosswalk detection missions is respectively to demonstrate the ability to detect the status of traffic lights and crosswalk locations. The vehicle should stop or pass the crosswalk according to the traffic light status. The passenger pickup mission aims at detecting a passenger and stopping within five meters. In the overtaking mission, the vehicle must safely overtake a slow vehicle. In the school zone and sudden obstacle mission, the vehicle should not only comply with the speed limit but also perform an emergency stop that avoids accidents with unexpected obstacles. In the barrier mission, a vehicle should detect a barrier which blocks the road and stop within five meters of the barrier. In the construction site driving mission, a vehicle must pass a construction site that was not provided on the map. The objective of the split road mission is recognizing and following the direction of the traffic light for a split road. In the complex obstacle avoidance mission, the vehicle should find a drivable path without colliding with the complex obstacles. The parking mission aims at detecting a parking lot number and parking region; the vehicle then parks by itself in the corresponding parking position. If the vehicle succeeds in reverse parking, additional credit is provided as shown in Table I. The final winner of the AVC was determined by the sum of the travel time and the penalties for each mission.

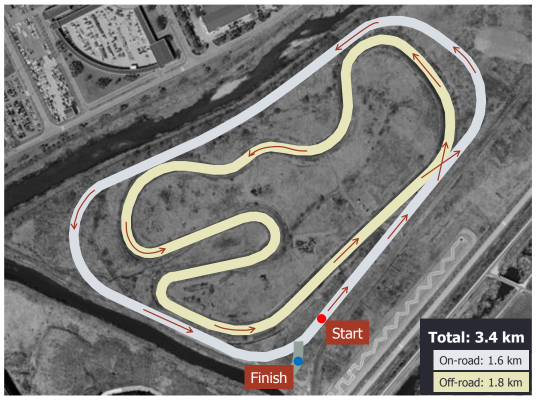


Fig. 1. The race track of 2012 AVC.

TABLE I
TIME PENALTIES AND CREDITS OF THE MISSIONS DEFINED IN THE AVC

Missions	Penalty (credit)
Traffic light and crosswalk detection	+2 min
Passenger pickup	+2 min
Overtaking	+2 min
School zone and sudden obstacle	+2 min
Barrier	+2 min
Construction site	+2 min
Split road	+2 min
Complex obstacle avoidance	+1 min
Parking (reverse)	+2 min (-2 min)

III. Autonomous Car *A1*: System and Algorithm

A. *System Overview of Autonomous Car A1*

Fig. 2 describes the autonomous car A1 and its sensor configuration. The vehicle platform was equipped an Electronics Stability Control (ESC) system. ESC is used to improve the vehicle dynamic stability by detecting the abnormal vehicle motion and controlling the vehicle motion using a brake system. In order to detect the abnormal vehicle motion, many types of on-board sensors are used for ESC, including wheel speed sensors, a steering wheel angle sensor, and a yaw rate sensor. The on-board ESC sensor information are shared through the in-vehicle network, Controller Area Network (CAN). Therefore, we can access the on-board sensor information of the ESC by connecting to CAN. A Motor Driven Power Steering (MDPS) system is used to control the steering, and the acceleration pedal signal is emulated for the acceleration. A DC motor that is mechanically connected to the brake pedal is used for braking. The gear shift lever is controlled by a motor to determine the direction of the vehicle.

Eight laser scanners are installed on A1 as shown in Fig. 2(b). Two multi-layer laser scanners (Ibeo LUX) that measure obstacles up to 200m away in optimal environment are mounted on the front bumper to detect distant objects. To detect the adjacent objects around the ego-vehicle, four single-layer laser scanners (LMS 151) are installed on each corner. Two single layer-laser scanners (LMS 291) that scan vertically to the ground are installed on the roof to detect barriers which are thin bars placed horizontally to the ground.

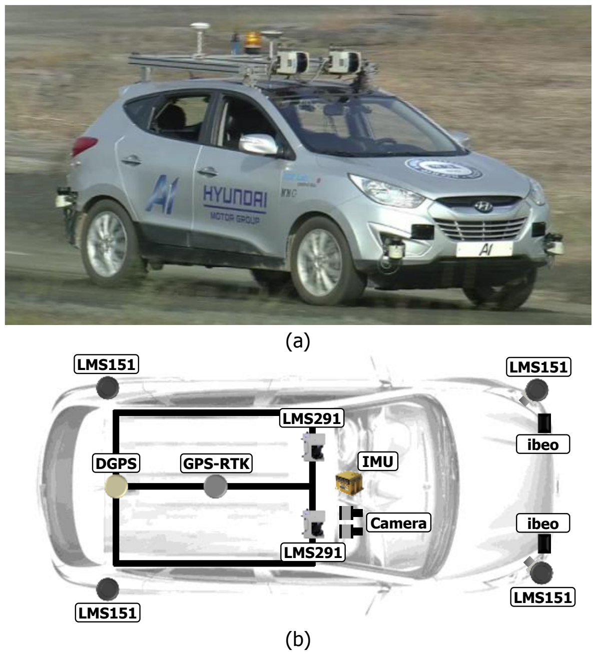


Fig. 2. Vehicle platform and sensor configuration of the A1.

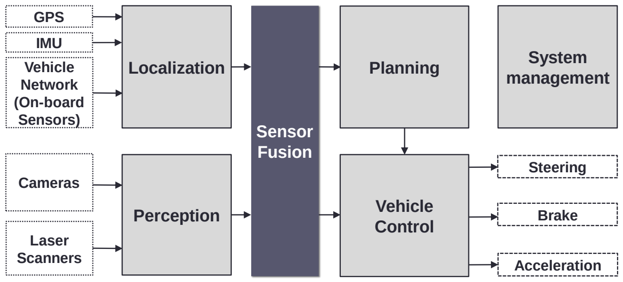


Fig. 3. Structure of the autonomous driving algorithms in the A1.

To detect and classify the mission objects, one color camera and three mono cameras are installed on the inside of the windshield. The two types of GPS receivers, a Real-Time Kinematics GPS (RTK-GPS) and a Differential GPS (DGPS), are equipped in the vehicle to measure the position of the ego-vehicle in a global coordinate frame. An Inertial Measurement Unit (IMU) which is located in the center of the vehicle is used to estimate the vehicle's dynamic motion.

B. Autonomous Driving Algorithm

In order to autonomously drive without human intervention, an autonomous car requires five basic functions: localization, perception, planning, vehicle control, and system management (Fig. 3). The localization is responsible for the estimation of the vehicle position, and the perception derives a model of the driving environment from multi-sensor fusion-based information. Based on the localization and perception information, the planning function determines the maneuvers of the autonomous car for safe vehicle navigation. The vehicle control function follows the desired command from the planning function by steering, accelerating, and braking the autonomous car. Finally, the system management supervises the overall autonomous driving system. Detailed descriptions of the autonomous driving algorithm of A1 is follow.

1) Localization

A localization system is an essential component of an autonomous car since autonomous cars find optimal paths and control vehicle motion only if the ego-vehicle position from the localization system is available. A GPS is widely used for the localization system because it directly provides a global position and the ego-vehicle's velocity. However, the raw data position of the GPS cannot be used for the autonomous driving system since the quality of the GPS position is significantly affected by satellite signal conditions. The accuracy, reliability, and continuity of measured GPS position data will rapidly deteriorate when GPS satellite signal conditions are unstable due to blockage and multipath.

A lot of previous research focused on the fusion of a GPS with additional information such as vehicle motion sensors (wheel speed sensors, gyro, accelerometer and magnetic sensors) [15, 16], environment perception data [17], and digital maps [18] in order to compensate for GPS vulnerabilities. The basic principle of an information fusion-based localization system is that GPS position errors are corrected by another information source such as vehicle motion constraints and correction data from matching the perceived landmark with a digital map.

In this paper, in order to cover the various driving conditions, an Interacting Multiple Model (IMM) filter-based information fusion system is used for the localization system [16, 19]. The localization system can adapt to changing vehicle dynamic characteristics under various driving conditions since the IMM filter selects the kinematics and dynamics model according to driving conditions. A GPS-bias correction algorithm was also applied to the localization system in order to improve the accuracy and reliability of the localization system [18].

2) Perception

The perception system offers information on surrounding environments using several types of sensor components such as cameras, radars, and laser scanners. A range sensor (radar and laser scanner) based perception system detects and tracks static and dynamic obstacles [20, 21], while a vision-based perception system recognizes various visual objects [22, 23]. Detected and recognized objects data are used for situation assessment of the autonomous driving system.

The perception system of A1 consists of a laser scanner-based moving object tracking algorithm and a vision-based object detection and classification algorithm. The object tracking system of A1 uses four LMS151s and two Ibeo LUXs to track moving vehicles around the ego-vehicle. In particular, an Integrated Probabilistic Data Association Filter (IPDAF)-based [24] tracking algorithm is implemented using raw-data. Tracked dynamic objects are integrated using a covariance-based track-to-track fusion algorithm to reduce the conflict detection from the six laser scanners [25].

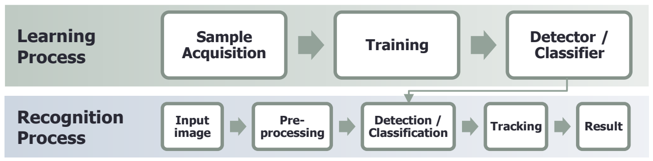
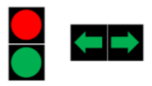
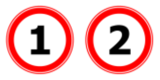
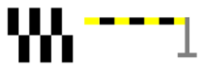


Fig. 4. Machine learning based scheme for object recognition.

TABLE II
SEVERAL VISUAL OBJECTS INCLUDING TWO TYPES OF TRAFFIC LIGHTS, TWO TYPES OF TRAFFIC SIGNS, AND PATTERN OBJECTS

Traffic Lights	Traffic Signs	Pattern Objects
		

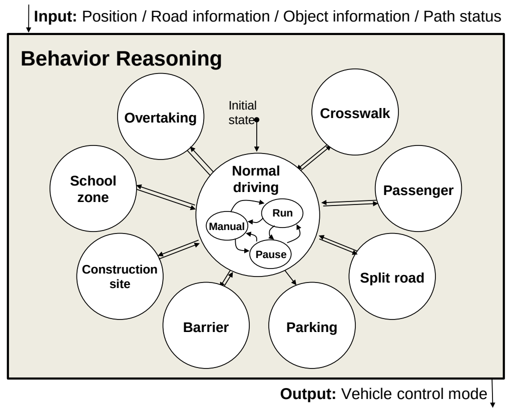


Fig. 5. Finite-state machine for behavior reasoning

There are several visual objects which should be detected and classified by the vision system: two types of traffic signs, two types of traffic lights, and the pattern objects as described on Table II. A machine learning-based scheme is employed to perceive the visual objects. The scheme consists of two main parts (Fig. 4).

We construct large training samples for each object at first in the learning process. The diversity of the training samples should be satisfied to establish the high performance of the detector or classifier; therefore, the training samples are obtained from various illuminations, poses and background conditions. Representative features are selected for each object in the training step; for instance, Haar-like features are used for traffic sign detection and a Histogram of Oriented Gradients (HOG) is employed for the traffic sign classification. Finally, machine learning algorithms are conducted such as Adaboost and Support Vector Machines (SVMs) to build detectors and classifiers [26, 27].

The input image is preprocessed for noise reduction and feature extraction in the recognition process. Next, the detector finds the location of the object in the image. The classifier identifies what types of objects are detected based on the detected image. Lastly, the object tracking method is applied to filter out false positives and integrate temporal recognition results. For instance, the Adaboost-based traffic sign detector finds where the circular traffic sign is in the image and then the circular traffic sign is categorized as the number one, the number two or others which are false positives by the SVM classifier with the HOG feature. The traffic sign is tracked by the Nearest Neighbor Filter (NNF) which estimates the position, width, and height of the traffic sign in the image [28]. The performance of the traffic sign recognition is over 95% under various lighting and weather conditions. From over 30,000 sample images, we evaluate that the recognition system performed well as long as the traffic sign image is not occluded with rain or saturated with direct light.

3) *Planning*

The planning system determines the maneuvers for autonomous cars. Planning algorithms for the autonomous cars can be divided into three stages in order to provide safe and reliable maneuvers under various driving situations [29, 30]: global routing, behavior reasoning, and local motion planning. Global routing finds the fastest and safest way to get from the initial position to the goal position. In this stage, a digital map management system and data searching algorithm are essential for fast routing [31]. Behavior reasoning assesses the driving situation and determines the overall behavior of the autonomous car based on the global route and perception information (Fig. 5) [32, 33]. The local motion can then be generated in the local motion planning stage based on the global route and the determined behavior. In the last stage, the generated local motion should avoid static and dynamic obstacle collisions for safe autonomous driving [34-36].

The A1's planning system focused on behavior reasoning and local motion planning since the road-map contains the global route of the track provided by the competition organizers. The map data represents the road geometry based on WGS84-type information with centimeter-level accuracy. The objectives of A1's behavior reasoning and local motion planning are to perform autonomous driving as well as to accomplish various missions in real time. The behavior reasoning executes a rule-based decision process based on Finite-State Machines (FSMs) for an overall driving strategy based on localization and perception information (Fig. 5). Incorporated in the decision process, the rule is pre-defined to follow traffic regulations (e.g. lane keeping, obeying traffic light, and keeping under speed limits) and accomplish various tasks.

In order to drive in various driving environments, the local motion planning is composed of two types of path planning algorithms: road map-based path planning and freeform path planning (Fig. 6) [5, 37, 38]. Road map-based path planning mainly works in normal situations such as lane driving, whereas the freeform path planning generates a complex path in unstructured environments such as roads under construction. The selected path planning algorithm with the behavior planner iteratively generates a safe and feasible path.

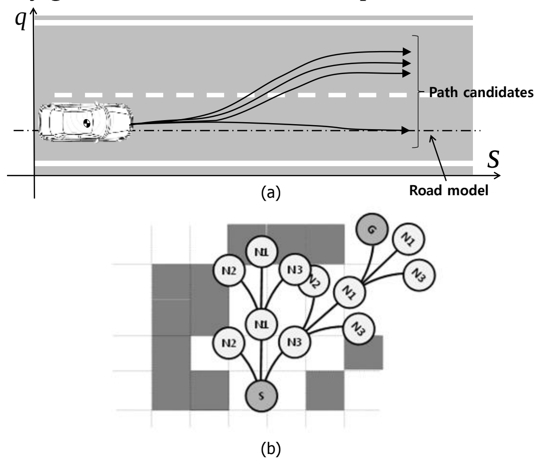


Fig. 6. (a) Lane keeping and changing path candidates in the road map-based path planning algorithm, and (b) graph structure-based path candidates in the freeform path planning algorithm.

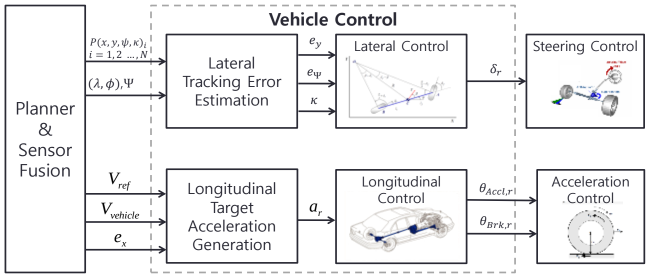


Fig. 7. Structure of vehicle control system.

4) *Vehicle Control*

Vehicle control is an essential function for guiding the autonomous vehicle along the planned trajectory. The vehicle control should be accurate for safe driving as well as robust under various driving conditions [39, 40]. In order to meet these requirements, the control system should be able to deal with several vehicle characteristics, such as nonholonomic constraints [41], vehicle dynamics [42], and physical limitations (e.g., constraints on the steering system and maximum allowable tire forces). In addition, a controller is required to solve the trade-off problem between tracking performance and ride comfort [43].

To deal with these vehicle characteristics in a practical way, the vehicle control system of A1 is divided into a lateral and longitudinal controller (Fig. 7). The lateral control algorithm assumes that the vehicle moves along Ackermann steering geometry [42]. Based on this assumption, the target steering angle is obtained from a lateral error of preview points in the generated path. In order to accurately track the path, the preview points and feedback gains are scheduled according to vehicle speed and path curvature information.

The longitudinal control algorithm derives the target position of the acceleration and brake pedals from reference inputs. In order to cope with various driving situations, the longitudinal controller is composed of three modes: speed control, distance control, and emergency stop. The speed control mode, which is based on the PID control law-based feedback controller and the vehicle powertrain model-based feed-forward controller, derives the target acceleration from the target and current vehicle velocity. Based on the target acceleration, the desired wheel torque is converted to manipulate the acceleration and brake pedals. In the distance control mode, an additional desired velocity calculation function is added for position control. The function generates the desired velocity using a position error between the current and target locations. When the autonomous car meets unexpected situations such as system faults and sudden obstacles, the emergency stop mode produces maximum deceleration for accident avoidance.

5) *System Management*

For the development and operation of an autonomous car, the system manager is essential for supervising the entire autonomous driving system. Basically, the system manager of A1 performs the following functions: Human Machine Interface (HMI), driving mode management, and fault management. The HMI consists of an operating interface (remote controller, and e-stop switch), and a display system that indicates the health status of the car, position, path, and surrounding object information. For the operation of A1, the driving mode is divided into three states: manual, run, and pause. In the manual mode, A1 is manipulated by the human; in the run mode, A1 drives by itself. If the operator pushes the emergency stop switch or a system fault occurs, the system mode is converted to the pause mode. The fault management system monitors the health status of all modules for safe driving. If the health status of the critical module for autonomous driving fails or does not update for a certain period of time, fail management algorithms, which are embedded in each module, determine the state of the system health as failed and convert the autonomous mode to the pause mode.

IV. Implementation of the Distributed System Architecture of A1 based on Software Development Process and Platform

The autonomous driving algorithm of A1 has been developed based on the proposed system development process and platform. The autonomous driving algorithms of A1 are first organized as software components. The software components are then mapped to the proper computing units. Finally, implementations of the software components are performed on the computing units based on the common software platform and in-vehicle network.

A. *Software Components Design*

Fig. 8 describes the designed software components and information flow of autonomous car A1. The A1 software architecture is composed of four parts: a sensor interface, autonomous driving algorithms, an actuator interface, and a development interface. Through the sensor interface, the various types of sensor data are entered into the autonomous driving algorithms. The GPS provides the global position, speed, and heading angle data for the positioning algorithm. The IMU and on-board sensors provide the dynamic information of the ego-vehicle to the estimation algorithm of the vehicle motion. The cameras and laser scanners measure information about the external environment around the ego-vehicle.

Based on the sensor information, the autonomous driving algorithms generate the control inputs of the vehicle actuators to drive the vehicle autonomously. The vehicle state estimation algorithm estimates the vehicle's dynamic states by integrating the motion sensor information with the vehicle system models. The estimates of the vehicle state are incorporated into the GPS data to more accurately and reliably estimate the ego-vehicle's position. The vision algorithms detect and classify the objects in the driving environment based on the image information from the color and mono cameras. The road barriers and static and dynamic obstacles are detected and tracked using the detection and tracking algorithms. The sensor fusion algorithm integrates all of the information from the perception algorithms and improves the accuracy and integrity of the primitive perception data. The planning algorithm uses the integrated perception data to determine the behavior and motion of the vehicle. The vehicle control algorithm calculates the control inputs from the vehicle actuator, such as steering, braking, and acceleration, to follow the desired behavior and motion from the planning algorithm.

The actuator interface conveys the actuator control input which is generated from the autonomous driving algorithm to the low-level controllers for steering, braking and acceleration. The development interface provides the safety and debugging functions for developing the autonomous system, including the emergency stop, wireless stop, driving mode selection, and monitoring.

The information flows between each software component are described with a VFB, as shown in Fig. 8. There are two types of communication between each software components: client-server and sender-receiver. At the client-server interface, the server provides the information to clients when the clients request the service. At the sender-receiver interface, the sender provides the information to the receiver without any request. Since the information flows are abstracted with the VFB, the software component designer does not need to be concerned with the constraints of the computing hardware and network. The designer can concentrate only on the design of the functional aspects of the software components.

B. *Computing Unit Mapping*

The layout of the distributed computing units and results of the software component mapping are represented in Fig. 9. The software components of the autonomous driving algorithm and system interfaces are distributed into the various types of computing units. The computing units of A1 consist of two embedded industrial computers, a Rapid Controller Prototyping (RCP) Electronic Computing Unit (ECU), and 13 ECUs that are based on 32-bit Micro Controller Units (MCUs).

The mapping of software components into computing units is performed by an experienced system designer. The designer considers several of the constraints for mapping the software components.

The sensor fusion, planning, and vision software components are mapped into two embedded industrial computers (Intel(R) Core 2 Duo 1.50GHz with 2GB RAM using Windows 7) for several reasons. First, the software components require high computing power to run; consequently, the software components are mapped onto a high performance computer system. In addition, all of the software components cannot be assigned into one computing unit; therefore, the software components are divided into two groups in consideration of algorithm execution time and dependency (Table III). The second reason is the hardware and software dependency of component systems. Many commercial sensors and monitoring devices support general purpose operating system (GPOS) environments such as Windows and Linux. Most software development kits of sensors (such as device drivers and libraries) are supplied as the GPOS version. Furthermore, some specific libraries such as OpenCV and BOOST, which can allow to accelerate implementation of vision or planning application, are easy to use on the GPOS environments [44, 45].

In addition, the vehicle control software component is mapped in an RCP-ECU. The RCP-ECU has high real-time performance for time-critical systems such as a vehicle control algorithm and provides development flexibility obtained from a well-organized user interface between a model-based control algorithm and implementation. Lastly, steering and brake software components are placed in one ECU because the components should operate longitudinal and lateral control actuation simultaneously with real-time constraints. The ECU is installed next to the actuators to reduce the wire length between the ECU and the actuators as well as to obtain a fast response. Other software components are mapped in a similar way.

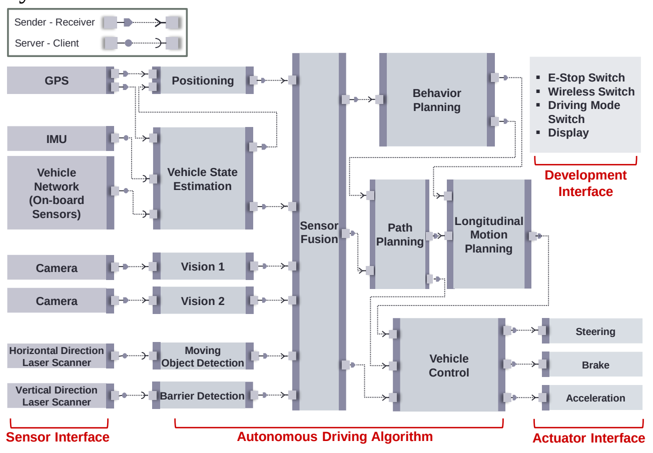


Fig. 8. Software architecture of autonomous car A1.

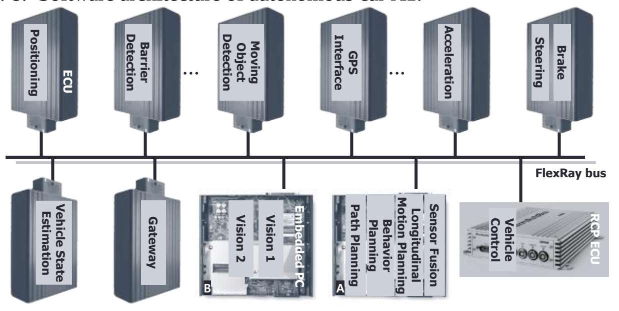


Fig. 9. Mapping of the autonomous driving algorithms and computing units.

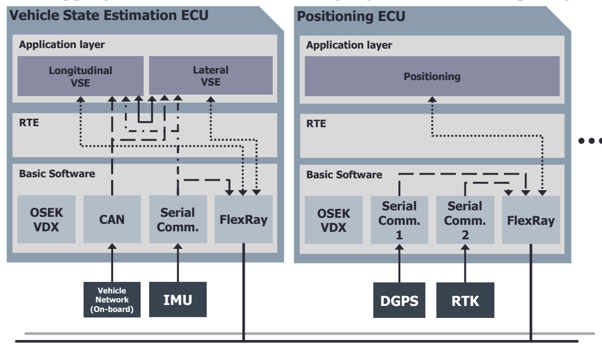


Fig. 10. Example of the implementation of software components using a common software platform.

TABLE III
EXECUTION TIME OF SOFTWARE ON INDUSTRIAL COMPUTER

Software Components	Execution Time (Full-resource used)	Assigned Embedded PC
Sensor fusion	10-15 ms	A
Longitudinal motion Planning	10-20 ms	A
Behavior Planning	5-15 ms	A
Path Planning	25-50 ms	A
Vision 1	15-25 ms	B
Vision 2	15-25 ms	B

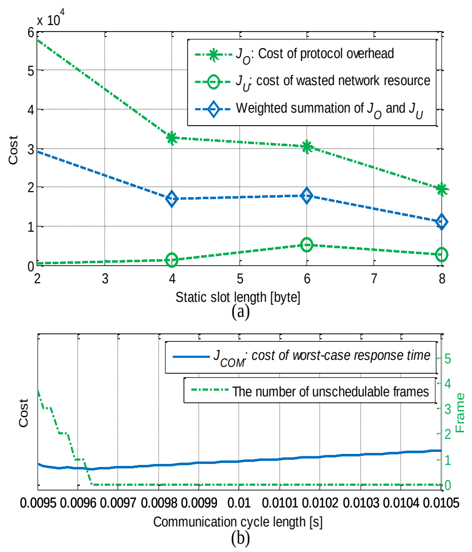


Fig. 11. Optimization results of FlexRay parameters.

C. *Implementation of Functions*

The software components assigned to the computing units are implemented using the proposed software platform and in-vehicle network. An example of the implementation of the software components for the vehicle state estimation and positioning is described in Fig. 10. Two ECUs are used for the implementation of the example software components, and each ECU is connected by FlexRay networks.

The software component of each function can be divided into several functional sub-components. The vehicle state estimation component of the example is divided into longitudinal and lateral vehicle state estimation. The functional components are independent of the computing hardware and network because there is a Runtime Environment (RTE). The RTE layer provides the standard interface between the application layers and basic software layer; therefore, the dependency between the software components and hardware can be minimized. The RTE also provides the scheduling interface between the application software component and the OS of the computing unit. The basic software layer provides the basic software components which are highly dependent on the computing unit and networks. In the example, the OSEK/VDX Real-Time Operating System (RTOS) is used for the scheduler, and serial communication is used for the sensor networks for the IMU, DGPS, and RTK-GPS. A CAN is used for accessing the vehicle network to obtain the on-board sensor information. FlexRay, which is the back-bone network of A1, is used to share the processing results with the other computing units.

As proposed in Part I, the FlexRay network design process has two parts: one is a network parameter configuration and the other is network message scheduling.

In the network parameter configuration step, the FlexRay network has more than 70 configuration parameters in order to operate properly, including cycle configuration parameters, start-up and wake related parameters, and clock correction related parameters. These parameters should be configured appropriately in order to take advantage of the time-triggered protocol. In the early stage of autonomous car development, the network parameter is heuristically configured to allow the modification of network design since a change of system configuration frequently occurs. We can determine which data should be transferred through the FlexRay network after the system configuration is fixed. The static slot length and communication cycle length can be obtained using the FlexRay network parameter optimization method based on the determined network data [46]. This method performs the optimization of the static slot length and communication cycle length to minimize the cost of protocol overhead J_O, wasted network resources J_U, and worst-case response time J_{COM}. Fig. 11 shows the optimization costs (J_O, J_U, and J_{COM}) of the FlexRay parameters of A1.

In order to optimize the static slot length in a FlexRay network, the optimization problem is formulated using J_O and J_U as follows:

$$\min \quad J_{\mathrm{STslot}} = J_O + J_U = (C_0 + x\cdot BSS)\cdot \sum_{k=1}^{n}\left\lceil \frac{msgST_k}{x}\right\rceil + \sum_{k=1}^{n}\left(x\cdot\left\lceil\frac{msgST_k}{x}\right\rceil - msgST_k\right) \qquad (1)$$

$$\text{subject to} \quad X = \{x \mid x = 2\cdot i,\ for\ i = 1,2,\dots,\max\left(\left\lceil\frac{msgST_k}{2}\right\rceil\right)\}, \qquad (2)$$
$$k = 1,2,\dots,n$$

where $msgST_k$ is the byte length of each ST messages, C_0 is the sum of the protocol-overhead length, including the header, trailer, TSS, FSS, FES, idle delimiter, action-point offset, and safety margin. BSS is the additional coding element for each byte of the frame. As described in the above optimization problem, the cost for the protocol overhead is closely related to the total number of ST frames. If we choose a long ST slot length x, the ST messages are divided into a smaller number of ST frames, so we can reduce the protocol overhead cost. However, if the ST slot length x is long, a large amount of unused network resource will result. Therefore, the protocol overhead and unused network resources must be considered concurrently. In Fig. 11(a), the two dotted green lines represent the costs of the protocol overheads, J_O, and wasted resources, J_U; in addition, the dotted blue line describes the weighted sum of J_O and J_U. From the results, the static slot length is selected as 8 bytes which is the minimal point of the cost.

The communication cycle length is optimized to minimize worst-case response time cost J_{COM} of the static and dynamic frames while satisfying the schedulability of all messages. The optimization problem is denoted as follows:

$$\begin{aligned} \min \quad J_{\mathrm{COM}} &= \sum_{i=1}^{n} R_{ST_i} \qquad (3)\\ &= \sum_{i=1}^{n}\left(C_{com.cycle} + C_i\right)\\ &= \sum_{i=1}^{n}\{(C_{ST} + C_{DYN} + C_{NIT}) + C_i\}\\ &= \sum_{i=1}^{n}\{(ST_{slot}\cdot n_s + d\cdot MS + 200\cdot MT) + C_i\} \end{aligned}$$

$$\text{subject to} \quad n_{ST_{fail}} = 0 \qquad (4)$$

where R_{ST_i} is the worst-case response time of ST frame i. C_{ST},

C_{DYN}, and C_{NIT} are the durations of the ST segment, DYN segment, and NIT, respectively, and C_i is the communication time of frame i. $n_{ST_{fail}}$ is the number of unschedulable ST frames. In the cost function J_{COM}, the R_{ST_i} is proportional to the length of the communication cycle, so the C_{ST} is determined by the duration of the ST slot ST_{slot} and the number of ST slots n_s, and the C_{DYN} is defined by the number of minislot d and the number of bits in a minislot MS. The C_{NIT} is assumed as 200 macrotick (MT). Since the communication cycle is determined by the summation of C_{ST}, C_{DYN}, and C_{NIT}, the optimal communication cycle is determined as the minimum communication cycle within there do not exist unschedulable ST frames. Fig. 11(b) shows the cost of the worst-case response time and the number of unschedulable frames. The results show that a communication cycle length over 9.634 ms is acceptable; consequently, we selected 10 ms as the communication cycle length in the FlexRay network of A1.

Network messages are scheduled to synchronize application software with the FlexRay network after the network parameters are optimized. The synchronization of the network messages can minimize network delays and make the delay predictable. This is a powerful advantage, especially for the safety related applications of autonomous cars because unexpected behavior caused by an unpredictable long network delay can be eliminated. The worst-case execution time of application software should be measured in order to synchronize the messages. The worst-case execution time can be found from the execution time measurements of several test cases. The test cases are generated by a code coverage analysis tool provided by MATLAB and Simulink. The measured worst-case response time is represented in TABLE IV. The measured worst-case execution time allows the network messages to be assigned to each static slot for the synchronization of application software.

Fig. 12 describes the synchronization results of the network message for the safety-related applications of A1 (TABLE IV). The vehicle state estimation ECU starts computing after receiving the sensor data which is from the on-board sensor and GPS. The FlexRay network is updated with the results after the calculations for the vehicle state estimation are completed. The positioning algorithm then uses the vehicle state estimation results for the position estimation and updates the FlexRay network with the position estimates. In the same manner, the vehicle control operates after receiving positioning information and generates control inputs for acceleration, braking and steering. Finally, the acceleration, braking and steering ECUs receive the control input and control each actuator.

Ethernet and CAN networks, which are event-trigger protocols, are also used for the communication system as the local sensor networks of A1. Point-to-point network topology between a sensor and computer is applied for the event-trigger network to minimize the effect of jitter in the event-trigger protocol.

D. Implementation Results

1) Development Process and Software Platform

The proposed software development process and platform has various advantages for the development of the autonomous car A1. The distributed system architecture could reduce the computational complexity of A1. When the development of A1 began, we used the centralized system architecture. Since the initial requirement was waypoint tracking in an obstacle free region, there was no problem with the computational resource. However, as requirements increased, such as moving obstacle tracking, traffic signal detection, and so on, the centralized system could not schedule all of the functions. In order to resolve this problem, we considered a more high-performance computational unit, but it could not resolve the computational complexity. In addition, it required additional costs and power sources. Therefore, a distributed system architecture was applied for decentralizing the computational load. Consequently, we reduced the computational complexity through decentralized computation, as well as saved costs and power sources.

The fault management system was applied to A1 in order to operate the autonomous car safely. Every computing unit in the A1 basically generates a life-signal for a health status check. Based on the signal, the fault management algorithm diagnoses the status and will back up the failed unit. By applying the fault management algorithm, the distributed autonomous driving system of A1 prevented some accidents due to negligence or software error.

The distributed system architecture of A1 was able to reduce the noise and cost from the wiring. Initially, the acceleration pedal position sensor in A1 was too far from the vehicle controller, so the wire between the sensor and controller was too long. The long wire length created noise in the measurement of analog signals, and caused degradation of the vehicle control performance. In order to resolve these problems, a measurement function for the acceleration pedal position sensor was transferred to another ECU located near the sensor, and the length of the wires was greatly reduced. As a result, not only was the noise problem resolved, but the stability of the controller was also increased. In addition, this reconfiguration work was not difficult due to the transferability of the proposed software architecture.

The proposed distributed architecture could support the development and testing being carried out simultaneously. In the development of A1, many developers were involved. Depending on the role of each development part, each individual person developed and tested their own software. In order to improve development efficiency, the individual software could be independently developed and tested by considering the functional and temporal characteristics.

The proposed development process and system platform enabled flexible system changes and extension of the system structure of A1. For instance, when the brake controller of A1 was developed, there were frequent hardware changes due to performance limitations. However, since the system development process is not dependent on the hardware, the development of the brake controller software was not influenced by the hardware change. As another example, there were several changes to the sensor layout in A1. When two laser scanners were added to detect rear objects, additional software was not required to be developed. The reason for this is that the existing software can be reused and modified easily.

With respect to the real-time performance, the proposed distributed system architecture exhibits more reliable computational processing than the centralized system architecture. Fig. 13 depicts the execution time of localization and vehicle control algorithms in each system architecture. Complete execution within an accurate period is an important factor for reliable autonomous driving in real-time since the localization and vehicle control algorithms are safety -critical functions for autonomous driving systems. In Fig. 13, the centralized platform generally has excellent computational performance in both localization and vehicle control operation. However, there exist several peak execution times which do not satisfy the execution-time constraints (Fig. 13). The reason for this is that the unexpected and sudden increase in the computation load for the other task affects the operation of the localization and control tasks.

The operation of localization and vehicle control in the distributed system consumes more execution time than the centralized case due to the computation power limitations of the implemented ECUs. However, the tasks of localization and vehicle control in the decentralized real-time embedded system are schedulable within the execution-time constraint since the distributed ECUs run relatively few tasks and operate with the OSEK-OS which is a standard RTOS for automotive embedded software [47, 48] (Fig. 13). The distributed system architecture is more suitable for hard real-time applications.

2) *FlexRay Network*

There are several advantages to using the FlexRay network-based distributed system. High bandwidth is one of most obvious advantages. The high bandwidth of the FlexRay network allows for sharing a large amount of information, compared to CAN, which is the most widely used for in-vehicle networks in the automotive industry. For example, the FlexRay network is able to send 211 messages every 10 milliseconds. However, the number of messages that can be transmitted by the CAN network is limited to 74 every 10 milliseconds. Since a total of 156 messages for A1 are used in the final network configuration, a periodic message transmission is possible without delay. Whereas, if the CAN network is used as the backbone network, a transmission delay of a message will occur and the overall system performance can be reduced, as shown in Table V.

The time-triggered scheme of the FlexRay network ensures exact periods by using the global time. In addition, since the time of the transmission and reception of network messages can be predicted, the application software can be synchronized with the global time. Otherwise, the event-triggered scheme of the CAN network cannot guarantee the periodic transmission of a message. As shown in Fig. 14, static frames in the FlexRay network are transmitted without delay while CAN messages are transmitted with period jitter. In addition, the period jitters of the CAN network differ according to the priority of a message. Therefore, when compared with CAN, the time synchronized application software of FlexRay can be expected to have a better performance.

TABLE IV
WORST-CASE EXECUTION TIME OF SAFETY RELATED APPLICATIONS

Application software	Worst-case execution time
On-board sensor	127 usec
GPS interfaces	2.38 msec
Vehicle state estimation	1.02 msec
Positioning	1.48 msec
Vehicle control	3.14 msec
Brake and steering	2.51 msec
Acceleration	3.16 msec

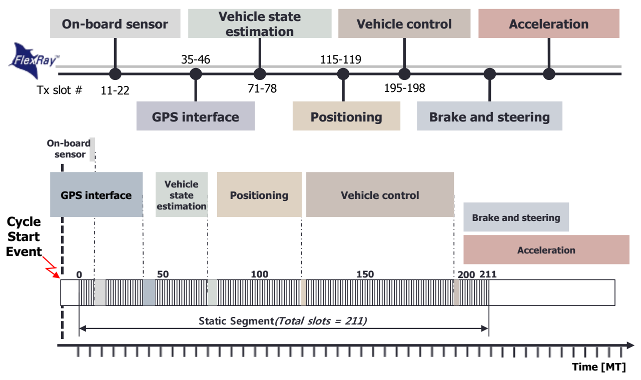


Fig. 12. Synchronization of safety applications of A1

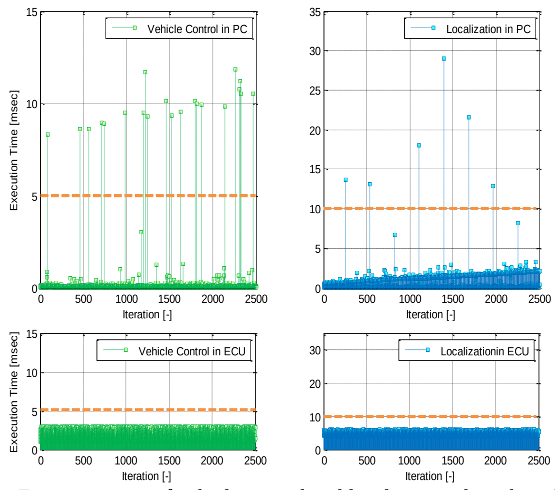


Fig. 13. Execution time of vehicle control and localization algorithm: (upper) centralized system architecture's cases with single PC, and (lower) distributed system architecture's cases with multiple ECUs.

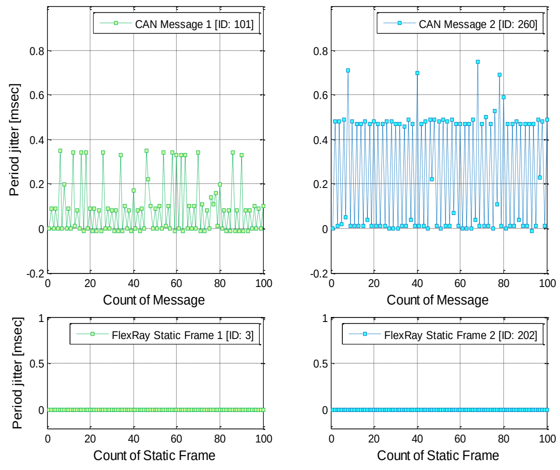


Fig. 14. Period jitters in the FlexRay and CAN Networks of A1.

TABLE V
COMPARISON BETWEEN THE FLEXRAY AND CAN NETWORKS OF A1

	FlexRay	CAN
Baud Rate	10 Mbps	500 kbps
Network Nodes	16	16
Messages	156	154
Bus Loads	73.93 % (Fixed bus load)	81.23% (100% peak bus load)
Others	Jitter free	Unpredictable latency, Jitter

V. DRIVING RESULTS OF A1 IN 2012 AVC

In order to evaluate the validity of proposed system architecture and development process, A1 drove itself on the race track and performed the missions in the 2012 AVC as shown in Fig. 1 and Table I. The pictures in Fig. 15 through Fig. 23 represent the mission results of A1. Each picture is a snapshot from the driving videos, which are taken from inside and outside of A1. As shown in these results, A1 not only drove the entire race track safely but also completed the missions perfectly. The maximum speed of A1 is up to 80 km/h during the final race. A1 won the championship of the 2012 AVC with a final time of 6 minutes 52 seconds with all missions completed. The final driving video is available at http://youtu.be/dzE2WReJeSM.

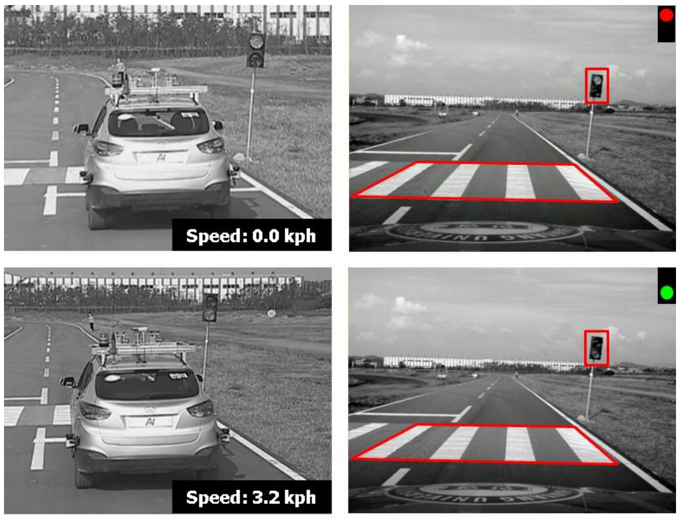


Fig. 15. Results of the crosswalk/ traffic light detection mission.

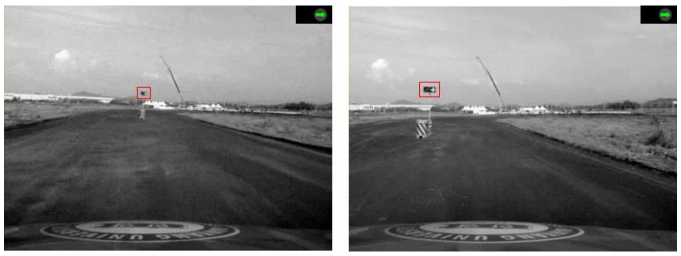


Fig. 16 Results of the split road mission.

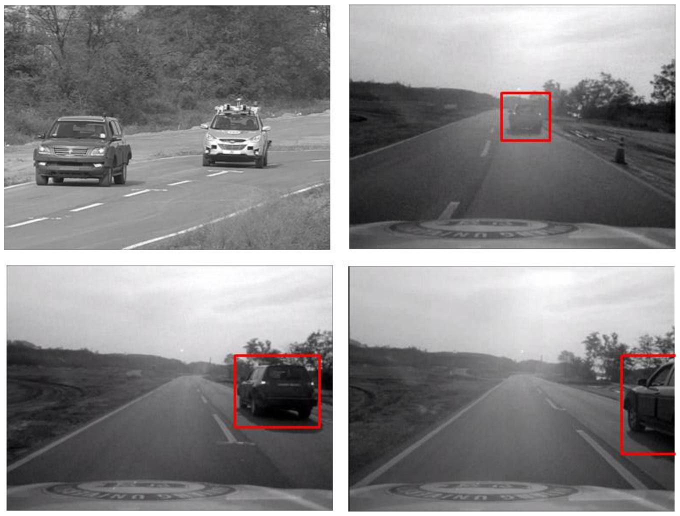


Fig. 17 Results of the overtaking mission.

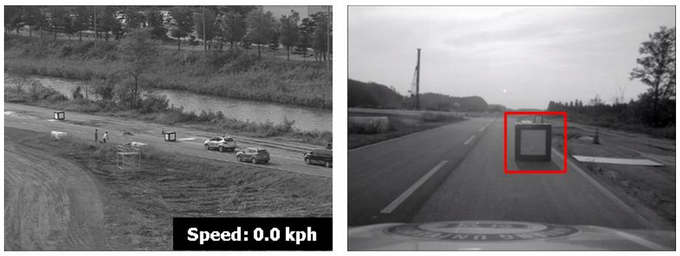


Fig. 18 Results of the school zone and sudden obstacle missions.

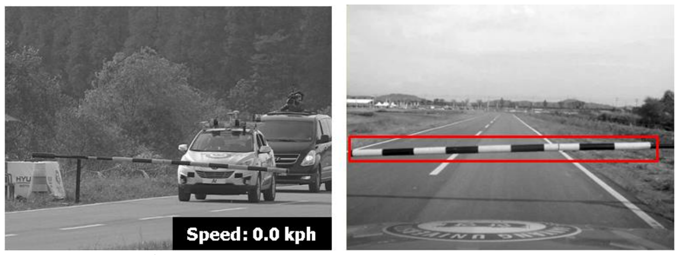


Fig. 19 Results of the barrier mission

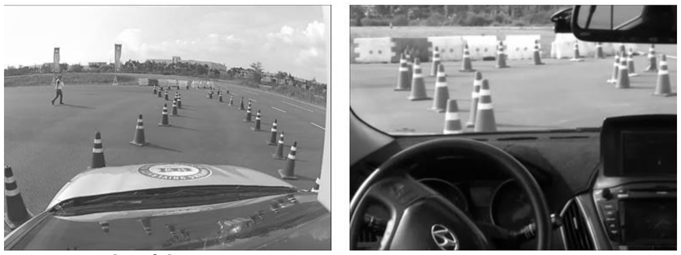


Fig. 20 Results of the construction site mission.

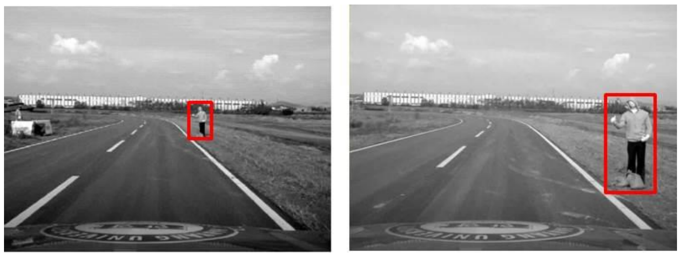


Fig. 21 Results of the passenger pickup mission.

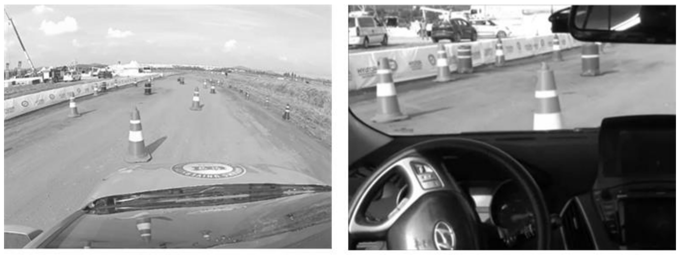


Fig. 22 Results of the complex obstacle mission

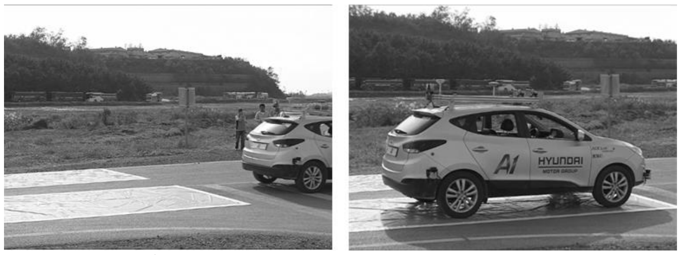


Fig. 23 Results of the parking mission.

VI. Conclusion

This paper (Part II) describes the implementation of the autonomous car that follows the development methodology proposed in part I.

In Part I, we addressed several advantages of the methodology from the following perspectives. First, the distributed system architecture can reduce the computational complexity of the entire system, guarantee fault-tolerant characteristics, and enhance the modularity of the system. Second, the development process provides comprehensive instructions for designing and integrating distributed systems of an autonomous car. Last, the layered architecture-based software platform (that originated from AUTOSAR) can improve the reusability, scalability, transferability, and maintainability of the application software.

In this paper, we present the case study of the development methodology by implementing the autonomous car. First, the five essential technologies of autonomous driving (localization, perception, planning, vehicle control, and system management) are briefly explained. Next, three steps (software components design, computing unit mapping, and implementation of functions) of the proposed development process are executed according to the instructions of the development methodology. In the procedure, several advantages of the distributed system are emphasized comparing the centralized system: computational load distribution, enhancement of system safety by cross-checking life-signal, noise and cost reduction from the wiring, and achievement of flexible system against to changes or extensions. The accomplishments of the autonomous car A1 that won the 2012 Autonomous Vehicle Competition in Korea prove the validity of the proposed development methodology.

However, the case study was performed only based on the specific computing platform, such as Windows and OSEK/VDX. The authors plan to extend the development process to adapt more software platforms such as Linux and ROS for developing the autonomous driving systems [49].

References

[1] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, *et al.*, "Stanley: The robot that won the DARPA Grand Challenge," *J. Field Robot.*, vol. 23, pp. 661-692, 2006.

[2] C. Urmson, C. Ragusa, D. Ray, J. Anhalt, D. Bartz, T. Galatali, *et al.*, "A robust approach to high-speed navigation for unrehearsed desert terrain," *J. Field Robot.*, vol. 23, pp. 467-508, 2006.

[3] M. Montemerlo, J. Becker, S. Shat, H. Dahlkamp, D. Dolgov, S. Ettinger, *et al.*, "Junior: The Stanford entry in the urban challenge," *J. Field Robot.*, vol. 25, pp. 569-597, 2008.

[4] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. N. Clark, *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *J. Field Robot.*, vol. 25, pp. 425-466, 2008.

[5] K. Chu, M. Lee, and M. Sunwoo, "Local Path Planning for Off-Road Autonomous Driving With Avoidance of Static Obstacles," *IEEE Trans. Intel. Transport. Systems*, vol. 13, pp. 1599-1616, 2012.

[6] J. Han, D. Kim, M. Lee, and M. Sunwoo, "Enhanced Road Boundary and Obstacle Detection Using a Downward-Looking LIDAR Sensor," *IEEE Trans. Vehichular Technol.*, vol. 61, pp. 971-985, 2012.

[7] K. Jo, M. Lee, D. Kim, J. Kim, C. Jang, E. Kim, *et al.*, "Overall Reviews of Autonomous Vehicle A1 - System Architecture and Algorithms," presented at the 8th IFAC IAV, Gold Coast, Australia, 2013.

[8] J. Kim, K. Jo, D. Kim, K. Chu, and M. Sunwoo, "Behavior and Path Planning Algorithm of Autonomous Vehicle A1 in Structured Environments," presented at the 8th IFAC IAV, Gold Coast, Australia, 2013.

[9] K. Jo, D. Kim, J. Kim, C. Jang, and M. Sunwoo, "Development of Autonomous Car - Part I: Distributed System Architecture and Development Process," *IEEE Trans. Ind. Electron.*, vol. 61, pp. 7131-7140, 2014.

[10] H. Heinecke, K.-P. Schnelle, H. Fennel, J. Bortolazzi, L. Lundh, J. Leflour, *et al.*, "AUTomotive Open System ARchitecture - An Industry-Wide Initiative to Manage the Complexity of Emerging Automotive E/E-Architectures," presented at the SAE Tech. Paper, 2004.

[11] W. Ruyi, L. Hong, Y. Min, W. Jinbo, and Y. Yuhao, "A hierarchical modeling method for AUTOSAR software components," in *IEEE Int. Conf. ICCET*, 2010, pp. V4-184-V4-188.

[12] D. Kum, G.-M. Park, S. Lee, and W. Jung, "AUTOSAR migration from existing automotive software," in *IEEE Int. Conf. Ctrl., Automation and Sys.*, 2008, pp. 558-562.

[13] S. Bunzel, "AUTOSAR–the Standardized Software Architecture," *Informatik-Spektrum*, vol. 34, pp. 79-83, 2011.

[14] "AUTOSAR Documents (V4.1)," *AUTOSAR Consortium*, vol. http://www.autosar.org/, 2013.

[15] I. Skog and P. Handel, "In-car positioning and navigation technologies: a survey," *IEEE Trans. Intel. Transport. Systems*, vol. 10, pp. 4-21, 2009.

[16] K. Jo, K. Chu, and M. Sunwoo, "Interacting multiple model filter-based sensor fusion of GPS with in-vehicle sensors for real-time vehicle positioning," *IEEE Trans. Intel. Transport. Systems*, vol. 13, pp. 329-343, 2012.

[17] I. P. Alonso, D. F. Llorca, M. Gavilan, S. A. Pardo, M. A. Garcia-Garrido, L. Vlacic, *et al.*, "Accurate Global Localization Using Visual Odometry and Digital Maps on Urban Environments," *IEEE Trans. Intel. Transport. Systems*, vol. 13, pp. 1535-1545, 2012.

[18] J. Kichun, C. Keonyup, and S. Myoungho, "GPS-bias correction for precise localization of autonomous vehicles," in *IEEE IV*, 2013, pp. 636-641.

[19] M. Gwak, K. Jo, and M. Sunwoo, "Neural-network multiple models filter (NMM)-based position estimation system for autonomous vehicles," *Int. J. Auto. Tech.*, vol. 14, pp. 265-274, 2013.

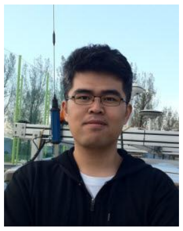
[20] K. Yeonsik, R. Chiwon, S. Seung-Beum, and S. Bongsob, "A Lidar-Based Decision-Making Method for Road Boundary Detection Using Multiple Kalman Filters," *IEEE Trans. Ind. Electron.*, vol. 59, pp. 4360-4368, 2012.

[21] Z. Huijing, S. Jie, Z. Yipu, X. Junqiang, C. Jinshi, Z. Hongbin, *et al.*, "Detection and Tracking of Moving Objects at Intersections Using a Network of Laser Scanners," *IEEE Trans. Intel. Transport. Systems*, vol. 13, pp. 655-670, 2012.

[22] L. Feng, D. Xiangxu, B. M. Chen, L. Kai-Yew, and T. H. Lee, "A Robust Real-Time Embedded Vision System on an Unmanned Rotorcraft for Ground Target Following," *IEEE Trans. Ind. Electron.*, vol. 59, pp. 1038-1049, 2012.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2015.2410258, IEEE Transactions on Industrial Electronics

[23] H. Yonghui, Z. Wei, and W. Long, "Vision-Based Target Tracking and Collision Avoidance for Two Autonomous Robotic Fish," *IEEE Trans. Ind. Electron.*, vol. 56, pp. 1401-1410, 2009.
[24] D. Musicki, R. Evans, and S. Stankovic, "Integrated probabilistic data association," *IEEE Trans. Automatic Ctrl.*, vol. 39, pp. 1237-1241, 1994.
[25] A. Houenou, P. Bonnifait, V. Cherfaoui, and J.-F. Boissou, "A track-to-track association method for automotive perception systems," in *IEEE IV*, 2012, pp. 704-710.
[26] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," San Diego, CA, 2005, pp. 886-893.
[27] P. Viola and M. J. Jones, "Robust Real-Time Face Detection," *Int. J. Comput. Vis.*, vol. 57, pp. 137-154, 2004.
[28] X. Rong Li and Y. Bar-Shalom, "Tracking in clutter with nearest neighbor filters: analysis and performance," *IEEE Trans. Aerospace and Electronic Systems*, vol. 32, pp. 995-1010, 1996.
[29] L. Li and F. Y. Wang, *Advanced motion control and sensing for intelligent vehicles*: Springer, 2007.
[30] R. Kala and K. Warwick, "Motion planning of autonomous vehicles in a non-autonomous vehicle environment without speed lanes," *Eng. App. of A.I*, vol. 26, pp. 1588-1601, 5// 2013.
[31] T. Ching-Chih, H. Hsu-Chih, and C. Cheng-Kai, "Parallel Elite Genetic Algorithm and Its Application to Global Path Planning for Autonomous Robot Navigation," *IEEE Trans. Ind. Electron.*, vol. 58, pp. 4813-4821, 2011.
[32] T. Bucher, C. Curio, J. Edelbrunner, C. Igel, D. Kastrup, I. Leefken, *et al.*, "Image processing and behavior planning for intelligent vehicles," *IEEE Trans. Ind. Electron.*, vol. 50, pp. 62-75, 2003.
[33] C. Baker and J. Dolan, "Street smarts for boss," *Robotics & Automation Magazine, IEEE*, vol. 16, pp. 78-87, 2009.
[34] N. Sudha and A. R. Mohan, "Hardware-Efficient Image-Based Robotic Path Planning in a Dynamic Environment and Its FPGA Implementation," *IEEE Trans. Ind. Electron.*, vol. 58, pp. 1907-1920, 2011.
[35] Y. Kuwata, S. Karaman, J. Teo, E. Frazzoli, J. P. How, and G. Fiore, "Real-Time Motion Planning With Applications to Autonomous Urban Driving," *IEEE Trans. Ctrl. Sys. Technol.*, vol. 17, pp. 1105-1118, 2009.
[36] S. Glaser, B. Vanholme, S. Mammar, D. Gruyer, Nouvelie, x, *et al.*, "Maneuver-Based Trajectory Planning for Highly Autonomous Vehicles on Real Road With Traffic and Driver Interaction," *IEEE Trans. Intel. Transport. Systems*, vol. 11, pp. 589-606, 2010.
[37] K. Chu, J. Kim, and M. Sunwoo, "Distributed system architecture of autonomous vehicles and real-time path planning based on the curvilinear coordinate system," *SAE Tech. Papers*, 2012.
[38] J. Kim, K. Jo, K. Chu, and M. Sunwoo, "Road-model-based and graph-structure-based hierarchical path-planning approach for autonomous vehicles," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, February 21, 2014 2014.
[39] V. Milanes, J. Villagra, J. Perez, and C. Gonzalez, "Low-Speed Longitudinal Controllers for Mass-Produced Cars: A Comparative Study," *IEEE Trans. Ind. Electron.*, vol. 59, pp. 620-628, 2012.
[40] C. Lin, A. B. Rad, and C. Wai-Lok, "An Intelligent Longitudinal Controller for Application in Semiautonomous Vehicles," *IEEE Trans. Ind. Electron.*, vol. 57, pp. 1487-1497, 2010.
[41] G. O. A. De Luca, and C. Samson., "Feedback control of a nonholonomic car-like robot," *Robot Motion Planning and Control*, pp. 171–249, 1998.
[42] R. Rajamani, *Vehicle Dynamics and Control*: Springer, 2012.
[43] A. Hemami and M. Mehrabi, "On the steering control of automated vehicles," in *IEEE Int. Conf. ITSC*, 1997, pp. 266-271.
[44] G. Bradski, "{The OpenCV Library}," *Dr. Dobb's J. SW Tools,* // 2000.
[45] J. Siek, L. Q. Lee, and A. Lumsdaine, *The boost graph library: user guide and reference manual*: Addison-Wesley, 2002.
[46] I. Park and M. Sunwoo, "FlexRay network parameter optimization method for automotive applications," *IEEE Trans. Ind. Electron.*, vol. 58, pp. 1449-1459, 2011.
[47] O. Group, "OSEK/VDX Operating System Specification," ed, 2009.
[48] A. Zahir and P. Palmieri, "OSEK/VDX-operating systems for automotive applications," in *OSEK/VDX Open Systems in Automotive Networks (Ref. No. 1998/523), IEE Seminar*, 1998, pp. 4/1-418.
[49] C. Ainhauser, L. Bulwahn, A. Hildisch, S. Holder, O. Lazarevych, D. Mohr, *et al.*, "Autonomous Driving Needs ROS (ROS as Platform for Autonomous Driving Functions)," in *ROS Developer Conf.*, Stuttgart, Germany, 2013.



Kichun Jo (S'10 -M'14) received a B.S. in Mechanical Engineering in 2008, and Ph.D. degree in Automotive Engineering in 2014 from Hanyang University, Seoul, Korea.

His main fields of interest are autonomous driving system, information fusion theories, distributed control systems, real-time embedded systems, and in-vehicle networks. His current research activities include system design and implementation of autonomous cars.

Junsoo Kim (S'11) received a B.S. in Mechanical Engineering in 2008 from Hanyang University, Seoul, Korea, where he is currently working toward a Ph.D. in the Automotive Control and Electronics Laboratory.

His main fields of interest are vehicle control, decision theories, path planning algorithms, and real-time systems. His current research activities include behavior reasoning and trajectory planning of autonomous cars.

Dongchul Kim received a B.S. in Electronics and Computer Engineering, and a M.S degree in Automotive Engineering in 2008 and 2010, respectively, from Hanyang University, Seoul, Korea, where he is currently working toward the Ph.D. degree in the Automotive Control and Electronics Laboratory.

His current research interests are in detection and tracking of moving object and information fusion. His research activities include the design of in-vehicle networks and real-time embedded software.

Chulhoon Jang (S'13) received a B.S. in Mechanical Engineering in 2011 from Hanyang University, Seoul, Korea, where he is currently working toward a Ph.D. in the Automotive Control and Electronics Laboratory.

His main fields of interest are autonomous driving system, image processing, and machine learning. His current research activities include object detection and classification using multiple sensor fusion and situation assessment for urban autonomous driving.

Myoungho Sunwoo (M'81) received a B.S. in Electrical Engineering from Hanyang University in 1979, and M.S. in Electrical Engineering from the University of Texas at Austin in 1983, and a Ph.D. in System Engineering from Oakland University in 1990.

He joined General Motors Research (GMR) Laboratories, Warren, MI, in 1985 and has worked in the area of automotive electronics and control for 28 years. During his nine-year tenure at GMR, he worked on the design and development of various electronic control systems for powertrains and chassis. Since 1993, he has led research activities as a Professor with the Department of Automotive Engineering at Hanyang University. His work has focused on automotive electronics and controls (such as modeling and control of internal combustion engines, design of automotive distributed real-time control systems, intelligent autonomous vehicles, and automotive education programs).