Proceedings of the IEEE ITSC 2006
2006 IEEE Intelligent Transportation Systems Conference
Toronto, Canada, September 17-20, 2006

RELIABLE ARCHITECTURE OF AN EMBEDDED STEREO VISION SYSTEM FOR A LOW COST AUTONOMOUS VEHICLE

Hannes Hemetsberger, Jürgen Kogler, William Travis, Reinhold Behringer and Wilfried Kubinger

Abstract— **In the 2005 DARPA Grand Challenge, five vehicles completed a preset course of 210 kilometers through desert dirt roads, completely driven by onboard automatic systems. This major achievement was accompanied by great progress of other vehicles which participated too but did not complete the course due to various reasons. The automatic vehicle RASCAL is one example of these vehicles: with its on-board autonomous capabilities, it reached a distance of 25.7 autonomously driven kilometers. One of the sensors implemented on RASCAL was a low-cost embedded stereo vision system. This dependable embedded system was based on a DSP platform and employed a novel software framework to guarantee reliable operations in the desert. This paper provides an overview on that particular subsystem.**

I. INTRODUCTION

The 2005 DARPA Grand Challenge on Oct. 8 2005 followed 18 months after the first Grand Challenge competition in 2004 [2]. In that initial event, none of the participating autonomous vehicles drove further than 12 kilometers, due to a variety of technical failures. The preparations of those teams for the 2005 DARPA Grand Challenge were based on learning from these failure reasons and were targeted towards improving the performance. Also the organizers had learned and had improved the layout of the course, keeping the challenging parts for the end of the route. And the conditions for being accepted into the final run in the desert were set high: at the National Qualification Event (NQE), the vehicles had to demonstrate perception and avoidance of collisions with obstacles. As a result, the vehicles admitted into the competition displayed a high standard of capabilities, and three vehicles (from Stanford University and Carnegie Mellon University) managed to complete the course within the given time limit of 10 hours on the first day [6], while two other vehicles also managed to complete the course the next day after an overnight stop, one of them still being within that 10-hour time frame.

The team SciAutonics/Auburn originally emerged from a collaboration between SciAutonics, LLC, and Auburn University. The Team included also engineers from the Conejo Valley and other areas of Southern California. It had participated in the 2004 Grand Challenge - its vehicle RASCAL had managed to complete only a distance of 1,2 kilometers of the complete course. However, in the 2005 Grand Challenge, the same vehicle achieved a total autonomously driven distance of 25,7 kilometers, more than 20-fold the previous performance. This performance improvement was a result of a complete re-write of the software, an improved system architecture, and an additional set of sensors such as an embedded stereo vision system from ARC Seibersdorf research.

This paper reports on the improvements that were done in order to achieve this performance in the 2005 DARPA Grand Challenge (DGC). In particular, the embedded stereo vision system will be described.

H. Hemetsberger, J. Kogler, and W. Kubinger are with ARC Seibersdorf research, Donau-City-Str. 1, A-1220 Vienna, Austria, hannes.hemetsberger@arcs.ac.at

R. Behringer is with Leeds Metropolitan University, Leeds, LS1 3HE, UK, R.Behringer@leedsmet.ac.uk

W. Travis is with Auburn University, Auburn Alabama 36849, USA, traviwe@auburn.edu

II. RASCAL

A. VEHICLE

The basis of the vehicle RASCAL (= Robust Autonomous Sensor-Controlled All-Terrain Land Vehicle) [4] is an upgraded military ATV Prowler, provided to the team by the company ATV Corporation (in Orange, CA). Its original base is a Yamaha ATV that is strengthened mechanically with metal tubing frame and robust shocks. In the preparation for the 2005 DGC, changes were made to this mechanical structure, except that a wooden platform was added on the front for easy mounting and removal of the sensor suite.

Fig. 1. The vehicle RASCAL at the DARPA Grand Challenge 2005.

Figure 1 shows the vehicle as it participated in the DGC, with the sensor suite mounted. Two generators provided a total of 4 KW electric power. The gas was stored in two tanks, providing sufficient fuel (of both generators and the vehicle engine) for a distance of at least 480 kilometers. The vehicle could be driven by a human driver - the servos for brake, throttle, and steering were implemented acting

1-4244-0094-5/06/$20.00 ©2006 IEEE

onto the existing human interface devices (paddle, steering wheel) [3].

Tests of autonomous driving could be performed on a piece of public land, provided by the City of Thousand Oaks.

B. SENSORS, COMPUTING NETWORK

The core of the autonomous system in RASCAL is the tight coupling between GPS and inertial sensing and the vehicle control (as described in section C). This module coupling in principle allowed autonomous operation without considering obstacles - that was the mode which was in fact employed in the 2004 Grand Challenge, as the sensing input was not yet properly implemented. Figure 2 shows the complete system architecture of the autonomous system implemented in RASCAL.

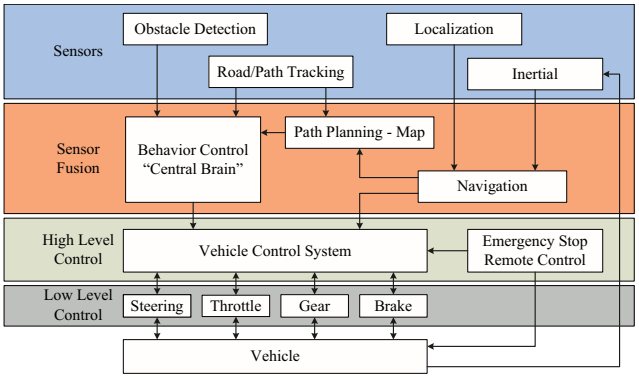

Fig. 2. System Architecture of RASCAL.

The other sensors were implemented as higher levels, providing optional input to the driving system. The overall goal was to achieve redundancy by allowing sensors of different type to cover the same regions (e.g. Lidar and stereo vision), to obtain robustness vs. failure of a single sensing mode.

The Lidar system was comprised of two vertically mounted SICK Lidars, providing surface tracking, and two horizontally mounted Lidars, providing object detection. The data from all sensors were first processed in modules attached to the data acquisition: they provided a conversion of the raw measurement data into compact object descriptors. In a central 2D representation, mimicking a birds-eye view, these detected objects were virtually plotted. This allowed a unified process to search this map for a suitable path, combining the intended course (way points and corridor, computed from the data given by DARPA) with the sensed environment.

The OS was Linux (2.6 kernel). The system was compiled as a set of independent running processes, allowing independent execution of internal loops. Communication between these modules was implemented using UDP - this allowed the modules being placed arbitrarily on a series of computers.

C. PATHPLANNER —— OPERATION

The initial RDDF (corridor definition) given to the team by DARPA had points spaced variably from a few meters out to hundreds of meters. The file was interpolated to generate a series of one meter spaced waypoints with desired speeds based on course conditions. A navigation algorithm was developed utilizing an Extended Kalman-Bucy Filter [10] to localize the vehicle on the course and to provide vehicle state information to other systems. The algorithm relied on a sensor suite consisting of a Rockwell Collins GIC-100 inertial measurement unit, a Navcom DGPS receiver with Starfire corrections, the ATV's speedometer, a Microstrain magnetometer, and a TCM2 magnetometer. This sensor suite provided position estimates as accurate as the DGPS unit when GPS was available. During an outage, the navigation algorithm could accurately dead reckon for several minutes with low error. Figure 3 displays the dead reckoning performance of the algorithm, where GPS was artificially removed just before the turn.

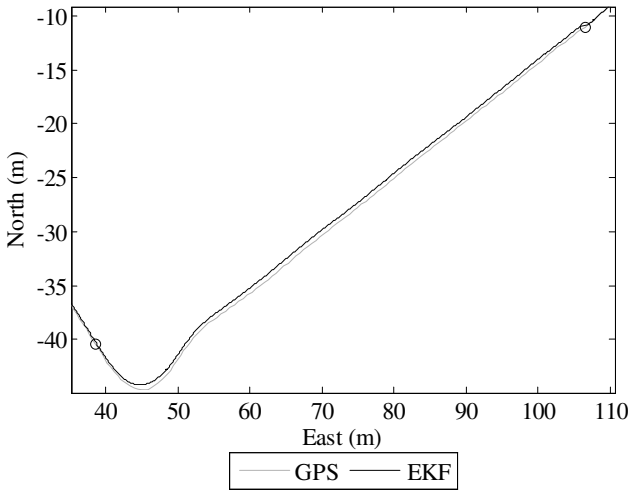

Fig. 3. Position estimation during a GPS outage

Obstacles were detected with two horizontally mounted Lidars, two vertically mounted Lidars, and an embedded stereo vision system. The path planner received information from the obstacle detection sensors and stochastically categorized their data to distinguish hazardous obstacles from the environment. The currently planned path was then compared against the list of obstacles. If RASCAL was on a path that would intersect an obstacle new paths were generated based on the vehicle's velocity. Paths that would lead the vehicle through an obstacle or out of bounds were discarded. If no suitable path was found, the vehicle was slowed to 1m/s.

The vehicle control module consisted of a path interface, speed control, and heading control. The path planner provided the path interface, which determined where the vehicle should drive based on the vehicle's position, orientation, and speed. The speed and heading control then used this information to determine the appropriate throttle and steering inputs. The speed controller was a PI, where the integrator was used to correct uncertainty in the DC gain. A PD controller was used to drive the heading error to zero. RASCAL's

controller was capable of operating at high speeds and was tested up to 17.8 m/s. For a vehicle the size of RASCAL, this is significant as it is proportional to a passenger vehicle travelling at highway speed.

III. STEREO VISION SENSOR

Normally, stereo vision systems are composed using several standard PC's. However, embedded solutions are not off-the-shelf products [8], that is mainly for the following two reasons. Firstly, computer vision algorithms (especially stereo ones) need a huge amount of processing power [5]. State-of-the-art embedded systems are not as powerful as modern PC systems. Secondly, the development on an embedded system is more time consuming to reach the same performance. But on the other hand, embedded solutions offer features which are a good reason for development and use such solutions for computer vision applications. Advantages are less power and space consumption and the benefit of a dedicated operating system (OS). In the following sections a concept for an embedded stereo vision system is presented. This includes the hardware used and the software framework realized for the usage at the DARPA Grand Challenge 2005. The vision algorithms which are used for obstacle and lane detection are described in more detail in [7].

A. CONCEPT

In comparison with other teams only two others used an embedded stereo vision system for obstacle detection. All the other teams processed the vision data on several PCs or notebooks which were connected to a cluster of computers. Due to the philosophy of the team to build a small and low cost vehicle, the hardware used was embedded and highly power aware. The whole electric power consumption of the developed system was less than 20 watts.

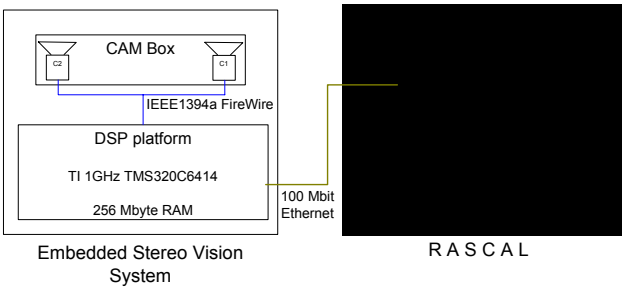

Fig. 4. Basic concept of the sensor

The basic concept of the embedded stereo vision system (ESVS) consists of two main parts, the embedded processing platform and the camera box. This box contains two bus-powered IEEE1394a FireWire [13] cameras. A box is required because of the desert area where the system is used. By driving in the desert the cameras must be protected against dust, heat, sunbeams and sand. The box should also be sealed up with window gasket to be sure that no dust gets inside the box. The IEEE1394a FireWire cameras are connected to the hardware platform and deliver images to the platform using the IEEE1394a FireWire bus [14]. The images are transferred with the isochronous transfer mode and are controlled with the DCAM standard V1.31 [1]. The extracted feature report of the detected lane borders and obstacles are sent in form of proprietary User Datagram Protocol (UDP) messages using the 100MBit Ethernet network [9]. Figure 4 depicts the system concept and the communication to RASCAL.

A very important design point was the specification of the field of view. This is affected by the mounting position, height and the used lenses for the cameras. The mounting position of the camera system is predefined by the mechanical construction of RASCAL. The requirements of the team was that the system shall operate in a range between 10m up to 35m in front of the vehicle. Therefore, the mounting position must guarantee that this requirement can be fulfilled. The main task was to find the best combination between camera lenses and mounting position to accomplish the stereo vision requirements. Furthermore the choice of the camera lenses and mounting position depends on the curve radius which the vehicle should be drive. In table I the parameters of the lenses used are shown.

Lenses	PENTAX C815B
Format	2/3 inch
Focuslength	8.5mm
Minimal object distance	0.2m
Horizontally Field of View β	56 degree
Vertically Field of View α	42 degree

TABLE I
DATA OF THE USED LENSES

The lenses used have a 2/3 inch format and the cameras have an 1/2 inch chip. Therefore, the parameters of the field of view presented in table I, for the 1/2 inch chip must be recalculated. The effective horizontally field of view for the 1/2 inch chip is 41.26 degree and for the vertically field of view is 31.54 degree. Figure 5 shows the final mounting position of the camera system. In this position all requirements including the operating range and the minimal curve radius are fulfilled.

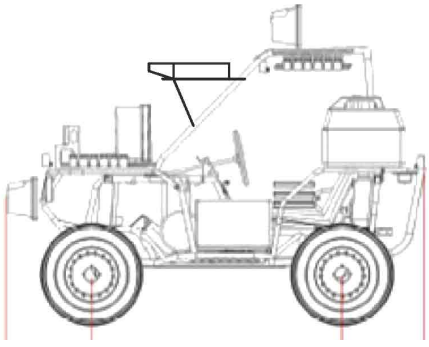

Fig. 5. Mounting position of the camera system

Due to the chosen lenses with the fixed field of views, a mounting height of 1.5m measured from the ground was

chosen. In addition, the baseline of 0.3m between the two cameras follows from the mounting position and the lenses used.

As mentioned above, the concept of the system contains of two parts, a camera box, which is mounted on a fixed position, and an embedded hardware. The ESVS should work stand alone, that means that after the system is mounted on the vehicle it only must be connected to the Ethernet network and and powered on. The system boots up and if any obstacles or lane borders are detected, this information is packed in the specified message format and sent to RASCAL brain.

B. HARDWARE

As mentioned above, the system consists of two main parts. The camera box was self made and self designed. The two cameras, which are mounted inside the box, are manufactured by BASLER. These cameras are bus-powered, have a resolution of 656 x 490 pixels and a frame rate of up to 60 frames per second (fps). These cameras are equipped with special infrared filters. These filters guarantee that the cameras are not influenced by the sunlight. The parameters of the lenses used are shown in Table I and manufactured by PENTAX. The hardware platform used is manufactured by STRAMPE SYSTEMS Electronics. This platform is equipped with a TEXAS INSTRUMENTS (TI) Digital Signal Processer (DSP) TMS320C6414 running at 1GHz [12]. A 133MHz 256 MByte notebook RAM module is used as external memory. This module is connected on one of the External Memory Interfaces (EMIF). A special feature of the platform are the changeable interfaces. Each interface, in our application a FireWire and an Ethernet module, is controlled from an own controller. These controllers are programmed at the startup. The platform is equipped with a main controller and this is responsible for the communication with the interface controllers.

With this high performance platform, a frame rate of 10fps, running both computer vision algorithms, should be achieved [7]. Figure 6 shows the system mounted on RASCAL at test drives somewhere in the Mojave desert.

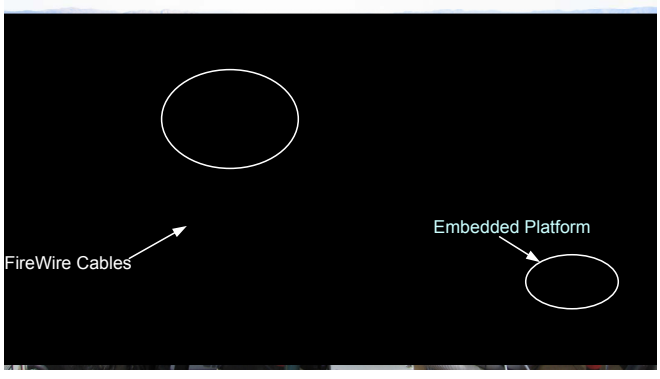

Fig. 6. Basic concept of the sensor

C. SOFTWARE

For the ESVS at the DGC a framework was developed. The framework was designed to support following aspects. The framework should:

- Be real time capable,
- guarantee that the images are processed with a constant frame rate,
- designed that it is easy to change tasks (because the framework should be re used in following projects),
- be a dependable solution (because of the race duration),
- be failsafe,
- and if a crash happens, the system should be able to react on this.

To fulfil all of these requirements the usage of an operating system (OS) was required. The OS should provide basic functions that are needed for the design of such a framework. Due to the usage of a TI DSP, the TI DSP/BIOS II as basic OS was used [11]. This is a preemptive operating system which supports the most basic functions like a task (TSK), semaphores and security functions. The framework sets up on the DSP BIOS and parts of the framework guarantee a stable and failsafe system.

It must be mentioned that the transfer of two images to the DSP takes about 40ms. In this time the DSP is either blocked or still waiting. To avoid this, an asynchronous grab command was used to give the CPU time to deal with another task(s).

The difficulty in this application was to design a system which guarantees a constant frame rate for the computer vision algorithms. That means that the period of the grabbed and processed images is constant. This is needed, because a variable framerate of the images would produce failures in the computer vision algorithms. For the design of the system we started to figure out which operations are needed. These are presented in the following listing.

- A part for the communication to RASCAL brain which sends the extracted features.
- A software watchdog which controls the system. This is needed because the DSP is not equipped with a hardware solution.
- A part for controlling the cameras and for grabbing the images.
- A part for each computer vision algorithm.
- A part for the initialization of the hardware.
- A part for the global time. This is needed because the hardware does not support the Network Time Protocol (NTP).
- A part for receiving data from RASCAL brain which are needed for the algorithms.

As mentioned above, a constant framerate is needed. Therefore, the usage of a time triggered OS would be helpful. Such a OS guarantees that each TSK is executed in a specified time slot. Several TSKs are connected to a virtual round which will be processed periodically. Due to this it is easy to achieve a constant framerate. In this application we can not use such an OS, because it does not exist any for

the DSP used. Therefore, a strategy was chosen to get nearly the same TSK flow like in a time triggered system. This was done using the features of the preemptive OS.

In such an OS the TSK with the highest priority gets the CPU time, excepting the TSK gets blocked by a semaphore. In this case the scheduler of the OS will give the CPU to the another,not blocked TSK, with second highest priority. This TSK runs as long as it gets blocked by a semaphore or a TSK with a higher priority would be woke up. Then the OS switches the TSK and the not blocked TSK with highest priority gets the CPU back.

This can be used to build the functionality of a time triggered task flow as follows. The system designer gives each TSK a priority in decreasing tendency. The TSK which should start working gets the highest. Each of these TSK also gets an own semaphore and when the system starts, each semaphore of the TSKs is taken and the TSKs gets in the blocked status (waiting on a semaphore). The idea now is that the first TSK is woken up by a semaphore with a periodically timer and this TSK resets all other semaphores, that means that each TSK change its status from blocked to ready. Due to the rules of a preemptive OS, the scheduler only give the CPU to these TSKs when no other with a higher priority is ready.

The first TSK (highest priority) wakes up all other TSKs, executes his job, and finally change his status into the blocked status. This means, that this TSK waits on the periodically semaphore. At this moment the second TSK starts working and when this TSK finishes his job it goes in the blocked status. This means, it waits until his semaphore is reset again, and so on. With this strategy a time slot based task flow like in a time triggered OS can be realized, because each task only gets the CPU when the task before finishes its job. This works well, if all of these tasks consume exactly the same processing time. Due to the usage of vision algorithms, this is given because sending data, receiving data and grabbing images consume the same processing time for each round. Due to this, the vision algorithms get the lowest priority and the round time is based on to the maximum processing time of the algorithm.

In this application this behavior was realized as follows. The TSKs, which are needed for the obstacle and lane border detection, are connected to a virtually round. Only, if all of these TSKs are executed, a feature report will be possible. Therefore, these TSKs gets priorities with decreasing tendency. This yields that the TSK with the highest priority starts working until it is finished and changes to blocked status, then the next TSK starts working and so on.

The TSK with the lowest priority finish the round. Figure 7 shows the basic task flow in the described application. To guarantee a constant period of this round, the *Timer TSK* is woken up periodically. This TSK resets all semaphores of the blocked TSKs and goes in the blocked status. Up to this, the *Grabbing TSK* starts working, and sends the asynchronous grab command using the FireWire bus to the cameras. The images are transferred to the interface and stored. Later, a Direct Memory Access (DMA) transfer copies the images

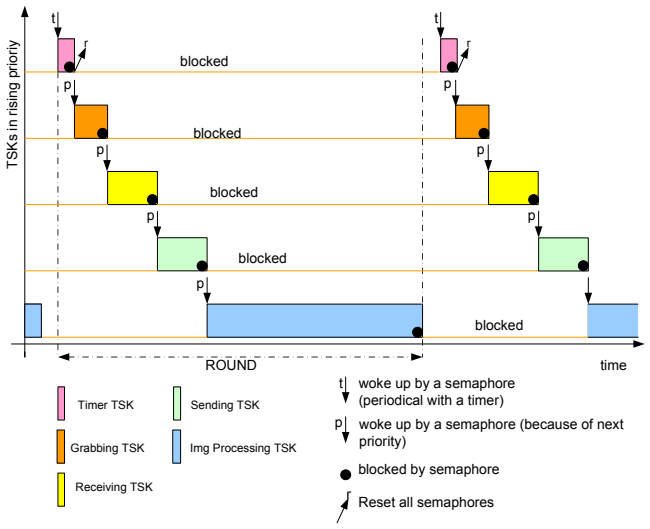

Fig. 7. Basic TSK work flow

into the external memory of the platform, without any usage of the DSP. After sending the grab command the TSK goes in the blocked status and the TSK with the next priority, the *Receiving TSK*, wakes up. This TSK is responsible for the transferring of the received message from the Ethernet interface to the DSP and for the analysis of this message. This TSK is followed by the *Sending TSK*, which sends the feature report to RASCAL brain and finally the *IMG Processing TSK*, which close the round. This TSK is responsible for the processing of the two computer vision algorithms, the obstacles detection and the lane border detection [7]. The round starts again after a constant period until a timer wakes up the *Timer TSK*.

This TSK system works stable and the fact that no other TSK directly depends on another, the tasks can be changed in an easy way. An existing TSK must only be deleted, a new inserted in the framework. This TSK only needs a correct priority and the framework is running stable again. An other advantage of this system is that the proven OS guarantees that the correct task will be executed. Thus, the programmer can not make mistakes by resetting the wrong semaphore, which could easily lead to a deadlock.

In the presented framework all TSKs which belongs to the virtually round, are called working tasks. In addition to these TSKs, there exists three other TSKs which are called system tasks. These are the *Watch TSK*, the *Init TSK* and finally the *Watchdog TSK*. The *Watch TSK* counts up the internal time, which is needed for a time stamp on the messages which are sent to RASCAL. The *Init TSK* initializes the hardware and the *Watchdog TSK* controls all TSKs. If any TSK crashes, the *Watchdog TSK* will notice this and perform a reboot of the system. All of the system TSKs have a higher priority than the working TSKs. The *Init TSK* only runs at the startup and starts the *Timer TSK* for the first time. The other two TSKs are woke up with a periodically timer. These TSKs need less processing time and so they hardly affect the processing

time of the processing TSKs. Figure 8 shows the task flow of all TSKs without the *Init TSK* and the needed processing time of the application. Real world tests before and during National Qualification Event (NQE) and DGC demonstrated, that the framework runs stable and failsafe.

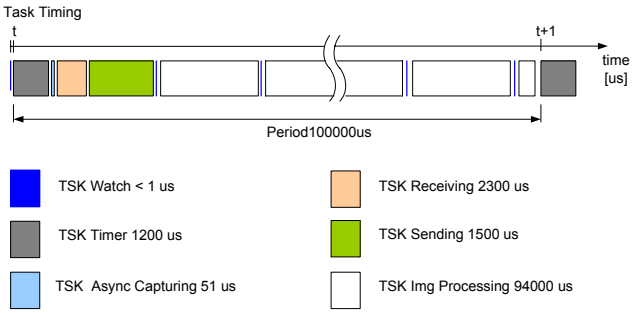

Fig. 8. The task flow of the DGC framework

The framework has a round of 100ms, all TSKs together needs 99ms. The two vision algorithms need the most processing time, the obstacle detection 40ms and the lane borders detection 49ms.

IV. RESULTS

RASCAL was the tenth vehicle that started at the 2005 Grand Challenge and completed the first leg without any problems. The vehicle started well and passed two other vehicles in the first few kilometers. However, problems developed and officials had to stop RASCAL because it lost the way and went off-road 25.7 kilometers into the course.

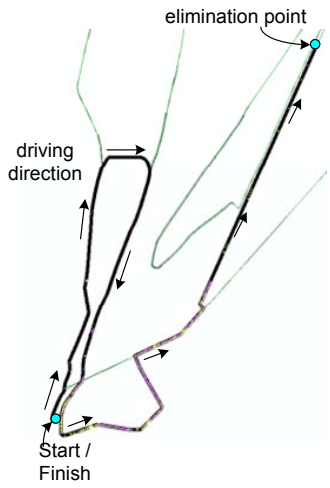

Fig. 9. The driven track of RASCAL

Figure 9 displays the course, which RASCAL completed. After analysis of the recorded data the cause of failure is fairly certain. The reason was an overheated USB hub: the hub failed as the vehicle was driving into its 17th mile on the course, disrupting the communication to the Lidar sensors and hereby disabling the recognition of obstacles. This led to a failure of other subsystems, and the vehicle had to be stopped by DARPA as it was about to collide with a power pole.

The presented ESVS was running all the time without any problems. The framework ran stable, three times the computer vision algorithms crashed due to a software flaw, but the software framework watchdog noticed this and performed a restart of the system. Until RASCAL was stopped, the sensor systems detected all obstacles which were placed in the course without any problems and RASCAL avoided all of them.

V. CONCLUSION AND FUTURE WORKS

The participation at the DARPA Grand Challenge was a great challenge for our team. The application area of this competition development demanded a robust hardware and software to achieve good results. The presented ESVS did a good job during the race and the system was tested under the hardest environmental conditions.

The presented ESVS will be enhanced to detect obstacles and lane borders in street traffic too. The framework is already reused in a new project, which deals with a low-cost embedded stereo vision sensor for indoor service robots.

REFERENCES

[1] 1394 Trade Association, *IIDC 1394-based Digital Camera Specification,* Version 1.31, 2002.

[2] Behringer, R. *The DARPA Grand Challenge - Autonomous Ground Vehicles in the Desert,* IFAC Symposium on Intelligent Autonomous Vehicles 2004, July 5-6, 2004, Lisbon, Portugal.

[3] Behringer, R., Sundareswaran, V., Gregory, B., Elsley, R., Addison, B., Guthmiller, W., Daily, R., Bevly, D.: *The DARPA Grand Challenge - Development of an Autonomous Vehicle,* IEEE Symposium on Intelligent Vehicles 2004. June 2004, Parma, Italy.

[4] Behringer, R., Travis, W., Daily, R., Bevly, D., Kubinger, W., Herzner, W., Fehlberg, V.: *RASCAL - An Autonous Ground Vehicle for Dessert Driving in the DARPA Grand Challenge 2005,* Proceedings of the ITSC2005 8th International IEEE Conference on Intelligent Transportation Systems, September 2005, Vienna, Austria.

[5] Gonzalez, R.C., Woods, R.E.: *Digital Image Processing;* Addison-Wesley Publishing Company, Reading, Massachusetts, 2. Edition, 1993.

[6] Goodwyn Th., Shipley D.: *Robots Conquer DARPA Grand Challenge,* Press Release, DARPA, October 8, 2005.

[7] Kogler, J., Hemetsberger, H., Alefs, B., Kubinger, W., Travis, W.: *Embedded Stereo Vision System for Intelligent Autonomous Vehicles,* IEEE Symposium on Intelligent Vehicles 2006, June 2006, Tokyo, Japan.

[8] Kopetz, H.: *Real-time systems: design principles for distributed embedded applications,* Kluwer Academic Publisher, 1997.

[9] Postel, J.: *RFC 768 - User Datagram Protocol,* 1980.

[10] Stengal, R.: *Optimal Control and Estimation,* Dover Publications, Mineola, NY, 2002.

[11] TEXAS INSTRUMENTS: *TMS320 DSP/BIOS User's Guide,* Nr. SPRU423B, 2002.

[12] TEXAS INSTRUMENTS: *TMS320C6414, TMS320C6415, TMS320C6416 Fixed-Point Digital Signal Processors,* Nr. SPRS146M, 2005.

[13] THE INSTITUTE OF ELECTRICAL AND ELECTRONIC ENGINEERS: *P1394 Standard for a High Performance Serial Bus, Specification,* V8.0v4, 2005.

[14] Yoshimoto, H., Arita, D., Taniguchi, R.: *Real-Time Image Processing on IEEE1394-based PC Cluster,* Proceedings of the 15th International Parallel and Distributed Processing Symposium, 2001.