2004-01-0459

Automatic Testing of an Autonomous Parking System using Evolutionary Computation

Oliver Bühler
STZ Softwaretechnik

Joachim Wegener
DaimlerChrysler AG, Research and Technology

ABSTRACT

The method of evolutionary functional testing allows it to automate testing by transforming the test case design into an optimization problem. For this aim it is necessary to define a suitable fitness function. In this paper for an autonomous parking system two different approaches for fitness functions are presented, which evaluate the quality of parking maneuver automatically. A numerical analysis shows, that the proposed area criterion supports a faster convergence of the optimization compared to the proposed distance criterion and that the proposed area criterion describes an efficient method to find functional errors in an automated way.

INTRODUCTION

Electronic control units in cars take over more and more complex tasks. Errors in the ECU's software can result in enormous costs. Therefore the aim is, to find as many errors as possible by testing. In practice dynamic testing is the mostly used analytical quality assurance method. A complete test is in practice not feasible, because of the huge number of possible test cases.

Up to now it was only possible to do the test case design manually, which required a considerable part of the project resources. In particular a directed search for functional errors during testing in an automatic way was not possible. The method of evolutionary functional testing makes it possible to automate test case design and allows a directed search for test cases to detect functional errors.

The method of evolutionary functional testing transforms the test case design into an optimization problem. A prerequisite for this is to evaluate the test results in an automated way. The evaluation is done by means of a fitness function, which assigns a numerical quality value to a test result.

This paper proposes two different approaches for the definition of fitness functions for an autonomous parking system. The defined fitness functions represent a quality metric and can evaluate a parking maneuver automatically. I.e. they return a numerical value which describes the quality of a parking maneuver. Both approaches are compared by numerical experiments for a prototype implementation of the autonomous parking system. The results show, that the proposed area criterion can identify critical parking maneuver better than the proposed distance criterion. The proposed area criterion supports a faster convergence of the test case optimization and provides a more efficient method to find faulty parking situations.

EVOLUTIONARY TESTING

Testing is aimed at finding errors in the system under test and giving confidence in its correct behavior by executing the system with selected input situations. A systematic test is divided into the core activities of test case design, test execution, monitoring and test evaluation as well as the activities of test planning, test organization and test documentation, which prepare for the test and accompany it [1].

Of all the test activities, test case design is assigned decisive importance. Test case design determines the type and scope and thus the quality of the test. If test cases relevant to the practical application of the system are forgotten, the probability to detect errors in the system sinks. Due to the central importance of test case design, a number of testing methods have been developed over the last decades to help the tester with the selection of appropriate test data. One important weakness of the testing methods available is that they cannot be automated straightforwardly. Manual test case design, however, is time-intensive and error-prone. The test quality does depend on the performance of a tester. In order to increase the effectiveness and efficiency of the test and thus to reduce the overall development and

maintenance costs for systems, a test should be systematic and extensively automatable. Both objectives are addressed by the method of evolutionary testing [2].

In order to transform a test aim into an optimization task a numeric representation of the test aim is necessary, from which a suitable fitness function for the evaluation of the generated test data can be derived. Depending on which test aim is pursued, different fitness functions emerge for test data evaluation. If, for example, the temporal behavior of an application will be tested, the fitness evaluation of the individuals of evolutionary testing will be based on the execution times measured for the test data [2]. For safety tests, the fitness values are derived from pre- and post-conditions of modules [3], and for robustness tests of fault-tolerance mechanisms, the number of controlled errors can form the starting point for the fitness evaluation [4]. Applications of evolutionary testing to structural testing result in different fitness functions again [5], [6], [7], [8].

If an appropriate fitness function can be defined, then the evolutionary test proceeds as follows. The initial population is usually generated at random. If test data has been obtained by a previous systematic test, this could also be seeded into the initial population. The evolutionary test could thus benefit from the tester's knowledge of the system under test. Each individual within the population represents a test datum with which the system under test is executed. For each test datum the execution is monitored and the fitness value is determined with respect to the defined test aim. Next, population members are selected with regard to their fitness and subjected to combination and mutation processes to generate new offspring. These are then also evaluated by executing the system under test with the corresponding test data. A new population is formed by combining offspring and parent individuals, according to the survival procedures laid down. From here on, the process repeats itself, starting with selection, until the test objective is fulfilled or another given stopping condition is reached.

EVOLUTIONARY FUNCTIONAL TEST OF THE AUTONOMOUS PARKING SYSTEM

THE AUTONOMOUS PARKING SYSTEM - As an automobile manufacturer, DaimlerChrysler is continuously developing new systems in order to improve vehicle safety, quality, and comfort. Within this context, prototypical vehicle systems are developed, which support autonomous vehicle parking - a function that might be introduced to the market in some years time.

The autonomous parking systems regarded in this paper are intended to automate parking lengthways into a parking space, like shown in Fig.1. For this purpose, the vehicle is equipped with environmental sensors, which register objects surrounding the vehicle. On passing along, the system can recognize sufficiently large parking spaces and can signal to the driver that a parking space has been found. If the driver decides to park in the vehicle can do this automatically.

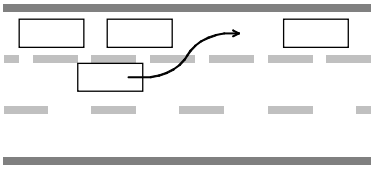

Fig.1: Functionality of an Autonomous Parking System

In Fig.2 the system environment for the autonomous parking system is shown. The inputs are sensor data, which contain information on the state of the vehicle, e.g. vehicle speed or steering position, and information from the environmental sensors, which register objects on the left and right hand side of the vehicle. For output the system possesses an interface to the vehicle actors, where the vehicle's velocity and steering angle will be set. The internal structure of the autonomous parking system is shown in Fig.3.

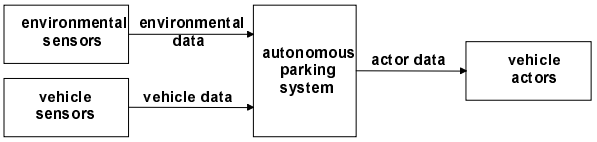

Fig.2: System Environment

The parking space detection processes the data from the environmental sensor systems and delivers the recognized geometry of a parking space if it has been detected to be sufficiently large. The parking controller component uses the geometry data of the parking space together with the data from the vehicle sensors to steer the vehicle through the parking procedure. For this purpose, velocity and steering angle are set for the vehicle actors.

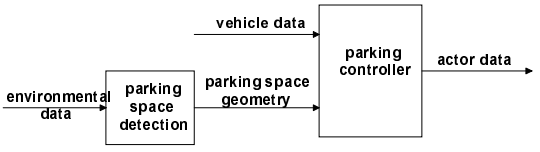

Fig.3: Sub-Components of Autonomous Parking System

APPLYING EVOLUTIONARY TESTING TO THE AUTONOMOUS PARKING SYSTEM - The automated parking system is a complex application. One reason for its complexity is the parking space detection, which has clearly to distinguish between drivable area and collision area. Another source of complexity is the navigation of the vehicle itself into the parking space. This has to be accomplished by controlling speed and steering angle and without touching the collision area. Entering the collision area involves a high probability of causing

damage to adjacent parking vehicles or objects. Thus it is a fundamental requirement of the autonomous parking system that the vehicle will not collide with other objects during the parking procedure. A system fault might cause considerable damage. For that reason exhaustive and efficient testing is essential before such a system might be released. This means that as many tests as possible must be performed in an efficient way.

Manual testing of the complete system is cost intensive and time consuming, because every test case comprises building up a park scenario with real cars and manual driving of each maneuver. Furthermore, performing a test in this way is difficult to reproduce, because the details of the test execution vary. In contrast, automated tests can perform a great number of test cases with less effort. Therefore, automated functional tests performed in a controlled simulation environment in addition to manual tests could form an important quality assurance measure.

Evolutionary functional testing provides a way of automating functional tests as a complete process. Instead of selecting the test cases manually, a search for interesting test cases is performed automatically. This is done by translating the test case selection into an optimization problem. This requires the solution of two problems. First, how to generate the test data and second how to evaluate the test results with the aid of a fitness function.

For the test data generation the possible input situations of the system under test are mapped to the search space. On one hand the mapping should keep the size of the search space as small as possible, on the other hand the mapping should be able to produce all possible input data for the system. If one considers the whole input range during design of the test data generator it does not mean that all test cases in this range will actually be tested, but it provides the possibility to generate any required test data. An appropriate model has to be designed for this purpose.

The evaluation of the test cases is carried out by the fitness function. In the automatic parking system, the fitness function calculates a numerical fitness value for the parking maneuver driven by the automatic parking system for the parking scenario generated. This fitness value represents the quality of the corresponding test case and intends to lead the evolutionary search into a direction of faulty input situations. The aim of the test is to find system faults and for that reason the fitness function is designed to assign good fitness values to parking scenarios which lead the system to enter the collision area or end up in an inadequate parking situation. Bad fitness values are assigned to scenarios which reach a good parking position with enough distance to the collision area.

TEST ENVIRONMENT - The test environment of the automatic parking system comprises the simulation environment, an evolutionary computation toolbox, an implementation of the fitness function and the test data generator which translates individuals into actual parking scenarios. The test object is the control unit of the vehicle with the implementation of the automated parking system inside.

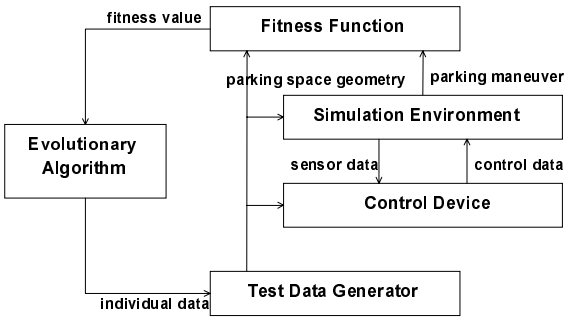

Fig.4: Design of the Test Environment

The GEA toolbox for Matlab [9] was used as implementation for the evolutionary algorithm. The simulation environment (built up on a Matlab R12.1 platform) simulates the properties of the vehicle and the surrounding environment. It runs with the control unit "in-the-loop" meaning that the simulation environment calculates the sensor data of the vehicle and presents it to the parking controller inside the control unit. The control unit processes this sensor data and reacts on it with control data for the simulation environment. This loop simulates a complete parking scenario. The parameters necessary for a simulation of a parking scenario, such as positions of the car and size of the parking space are outputs of the test data generator. After the simulation of a parking maneuver the fitness value is calculated by the fitness function and assigned to the generated individual.

DESIGN OF THE TEST DATA GENERATOR - The geometric data to characterize a parking space comprises six points P0 to P5, and is referred to as parking space geometry. The points define the border between the drivable and impassable area of the parking situation. The model for the generation of this parking space geometry is shown in Fig.5. It is a simplified model, because the borders of the parking space are always rectangular. The shape of the parking space can only vary in length and depth.

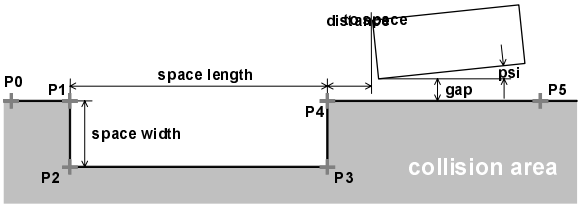

Fig.5: Model for Generation of Parking Space Geometry

This model takes the values of five independent variables and calculates from that the parking space geometry. The independent variables define length and width of the parking space, in addition to that at the starting position the distance to the parking space, the angle psi and the gap beside the vehicle on the right side.

DEFINITION OF FITNESS FUNCTIONS

This section describes the definition of two different strategies for the evaluation of the fitness of an parking maneuver. One strategy uses the distance between vehicle and collision area as measure for the evaluation of a fitness [10], the other strategy works with the area between vehicle and collision area.

Both strategies separate the parking space into two parts (1) collision with the precedent vehicle and (2) collision at the parking side. The possible movement of the vehicle into the parking space has limited degrees of freedom and thus if a collision with the precedent vehicle or the parking side happens the right rear edge or the right front edge of the car must be involved. From the position of the collision areas results, that to recognize a collision with the front vehicle, the lines between P3-P4 and P5-P4 have to be observed. To recognize a collision with the side, the line P2-P3 has to be observed. The definition distinguishes between observation of a corner, defined by three points, and the observation of an edge, defined by two points.

The fitness function in the test environment is intended to assess a parking maneuver and to assign an adequate fitness value to it. The fitness value should correspond to the degree of how good or bad the parking maneuver was. A good parking maneuver in this sense is when the vehicle does not collide with other parking cars. A bad parking maneuver in this sense is when other parking cars are damaged. The assigned fitness value for a parking maneuver should be the greater the better the maneuver is. Vise versa the value should become smaller for a more critical parking maneuver. The fitness value should become negative, when other cars are damaged during the maneuver.

DISTANCE CRITERION FITNESS FUNCTION - The distance criterion considers the closest distance between a vehicle edge and the collision border during the parking maneuver. In separate evaluations the smallest distance of the collision corner P3-P4-P5 and the smallest distance from the collision side P2-P3 are calculated. The following subsections describe, how the evaluation of the collision corner and collision side are done.

Evaluation of a Collision Corner - The collision corner is defined by three points P3-P4-P5. All distances will be calculated as polar coordinates with P4 as origin. The evaluation of the collision corner observes the section defined by P3-P4-P5 and the diagonal opposite section. Only the points within this sections are considered. As quality measure for the evaluation of the parking maneuver, the smallest distance between the vehicle path positions and the point P4 is taken.

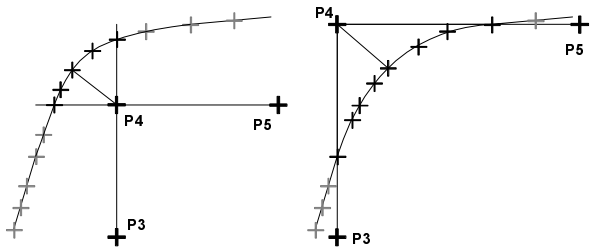

Fig.6: Selection of Smallest Distance

The value of the distance is positive, when the path is outside the collision corner. The value is set to zero, when the path crosses P4. The value is measured negative, when the path runs through the collision corner.

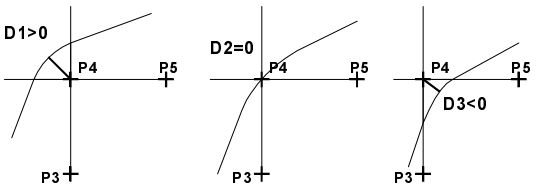

Fig.7: Signed Distance Values

This strategy shall ensure, that the more the path comes into the collision corner, the lower the assigned fitness value becomes. For different paths which continuously go into the collision corner the corresponding fitness values become continuously lower, like shown in Fig.7, where D1 > D2 > D3.

Evaluation of a Collision Side - The collision side is defined by the straight line between P2-P3. The distances are calculated between the line and the path positions. In this calculation only path points whose x-values are in the range within P2 and P3 are taken into account. The selection is done by comparing the x-coordinates from the path points with P2 or P3.

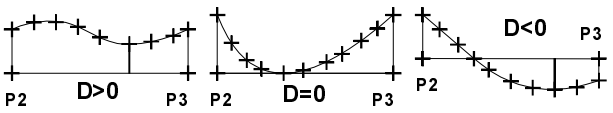

Fig.8: Distance from Line with Positive, Zero and Negative Value

The distance is calculated positive, when the path is above the line. The distance is set to zero, when the path touches the line. The distance is calculated as negative value, when the path runs through the collision area. From all calculated distance values, the minimum is taken as fitness value.

AREA CRITERION FITNESS FUNCTION - The area criterion fitness function considers the included area between the path of the vehicle and the parking geometry. Here in separate evaluations the included area in the collision corner and the collision side are calculated. The following subsections describe, how the evaluation of the collision corner and the collision side with the area criterion are done.

Evaluation of a Collision Corner - The area included between the corner lines and the vehicle path is taken as measure for the evaluation of the parking scenario. To calculate this area, it is separated into smaller segments, appropriate to the points of the vehicles path through the corner, like shown in Fig.9. The overall area A included in the collision corner is the sum of all segments together.

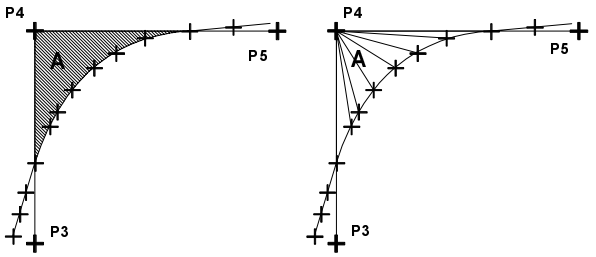

Fig.9: Separation of Included Area into Small Segments

For an effective and fast calculation of the fitness value, the area of each segment can be approximated by a triangle. When the distances between two points of the path T_n and T_{n+1} is significant smaller than the distance to P4 the angle $alpha_n$ is very small.

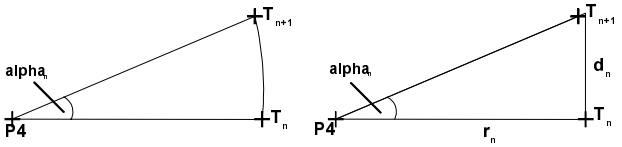

Fig.10: Approximation of Segment by a Triangle

For small angles of $alpha_n$, the ratio d_n to r_n is approximately the $angle_n$ in radiants. The triangle approximated area of one segment is

$$A_n = \frac{1}{2} \times r_n \times d_n \text{ and with } d_n \approx r_n \times (th_{n+1} - th_n)$$

the area of a segment can be calculated by

$$A_n \approx \frac{1}{2} \times r_n^{\,2} \times (th_{n+1} - th_n).$$

The angle th_n and the radius r_n for each path point T_n can be easily obtained, when the path positions are transferred from Cartesian coordinates into polar coordinates with P4 as origin of the coordinate system.

The overall area of the corner is the sum of all segments within the corner

$$A_{corner} = \sum A_n \ .$$

To lead the optimisation towards a collision, the corner P3-P4-P5 symmetrical to point P4 is taken into consideration, too. The path of the vehicle has to pass the point P4 and the aim of the optimisation is to bring that path into the collision area. The idea is, to rate an included area in the opposite corner as positive value and an included area in the collision corner as a negative value. When the vehicle path crosses through the corner point P4, the corresponding value is set to zero.

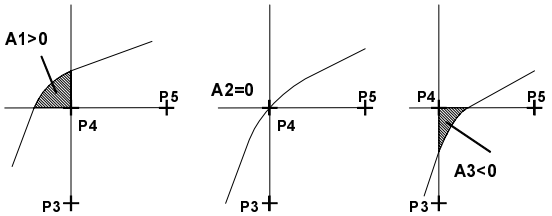

Fig.11: Signed Area Values

With that, the areas shown in Fig.11 become A1 > A2 > A3, when the vehicle path continuously shift into the collision corner. That leads to a gradual improvement of the fitness value, depending on how near the vehicle path passes the collision corner or crosses into it.

Evaluation of a Collision Side - The evaluation of a collision side takes as measure the included area between the vehicle path and the straight line P2-P3. The calculation of the area takes only those points into consideration, which have their x-coordinate in the range between P2 and P3. The calculation is done by approximation with the rectangles defined through the path positions. The area of each rectangle can be easily calculated by Δx and the distance between path and line Δy. With a sufficient number of path points in the range between P2 and P3, a good approximation of the included area can be achieved.

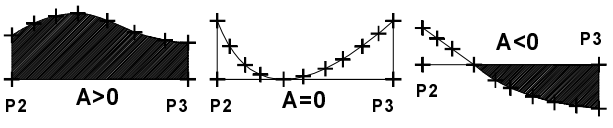

Fig.12: Area with Positive, Zero and Negative Value

To calculate the overall area, three different cases are differentiated, like shown in Fig.12. (1) when the path is above the line, (2) the path touches the line or (3) the path crosses the straight line into the collision area. In the first case, the included area is calculated as positive value, in the second case the value for the area is set to zero. In the third case the area below the line is calculated and taken as negative value. The distinction

into the three cases avoids that a bigger area above the line compensates a small area below the line and also that a touch of the line would be concealed by areas beside and could not be observed.

Compared to the distance criterion, which only takes the shortest distance into account, it seems to be more fair to consider the area between path and collision line. The aim of the experiments is to analyse and compare the results of both fitness functions. The issue is, how the different fitness functions influence the surfaces of their characteristic diagrams.

EXPERIMENTS

This section analyzes the results of the experiments with the test environment. Each subsection shows the result of one experiment and comprises two surfaces of the two different fitness functions. One figure is calculated using the area criterion and one figure is calculated using the distance criterion. In addition to that, the logarithmic representation of the absolute fitness values is shown, too. In each surface the values of two variables from the test data generation input vector are varied within a defined range and a defined number of samples. The values of the other three other variables are kept constant during that experiment. Each calculated point on a surface corresponds to a fitness value, which was calculated from one simulated parking maneuver. The height of the point on the surface represents the fitness value itself.

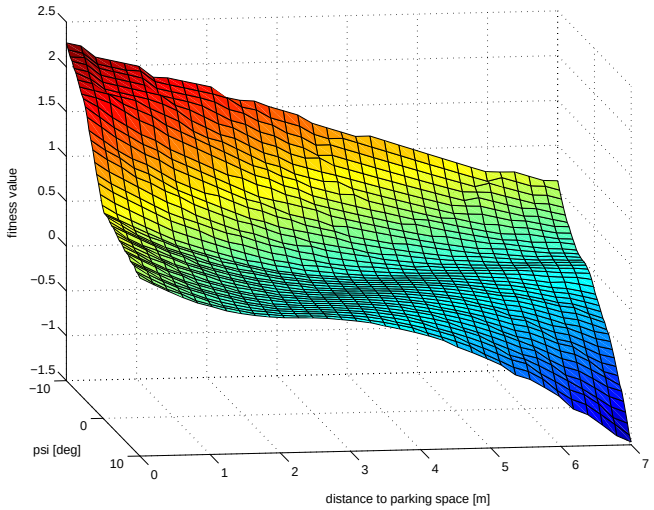

Fig.13: Fitness Values of Area Criterion as Function of Angle psi and Distance to Parking Space

DISTANCE TO THE PARKING SPACE AND PSI - In this experiment the variables dist2space and psi are varied, where the length and width of the parking space and the right gap beside the vehicle are kept constant. The length was set to 8.0 m, the width to 2.5 m and the right gap 0.7 m. The axis to the right shows the distance to the parking space in the range of 0.0 m to 7.0 m, with a resolution of 70 points. The axis to the depth shows the angle psi from +10 deg to –10 deg, with a resolution of 40 points.

Both functions return negative fitness values when angle psi reaches +10 deg and the distance to parking space goes towards 7 m. A considerable difference between the shapes of both surfaces is that for fitness values greater than zero the distance criterion returns constant small values where the area criterion descends its values towards the border to zero. This appears as a flat plateau in the distance criterion surface (Fig. 14) where the area criterion surface slopes (Fig. 13).

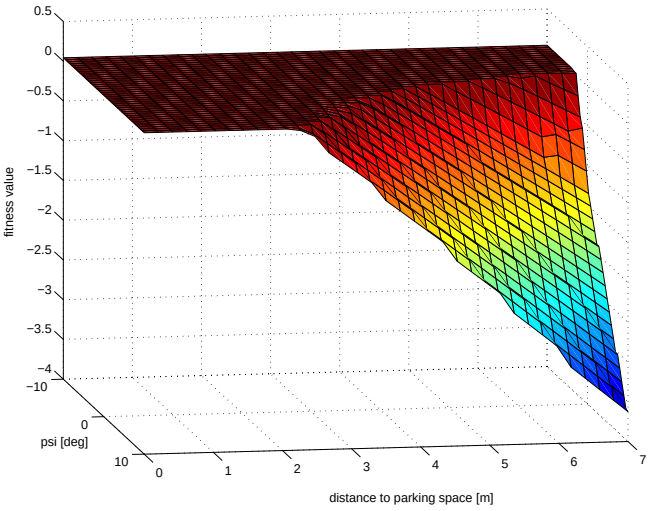

Fig.14: Fitness Values of Distance Criterion as Function of Angle psi and Distance to Parking Space

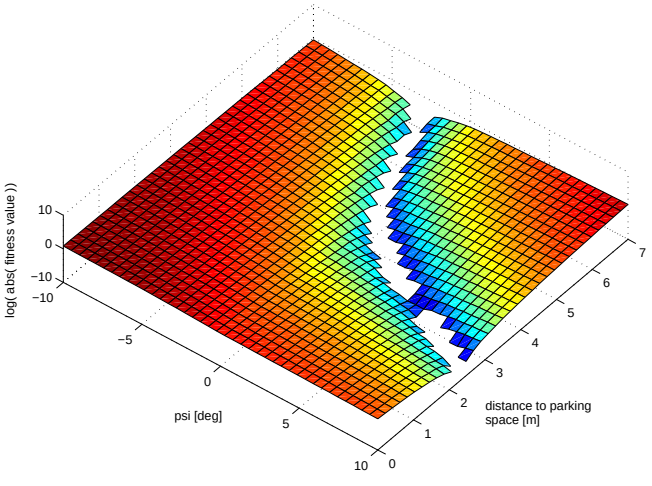

Fig.15: Logarithmic Representation of Absolute Fitness Values of Area Criterion

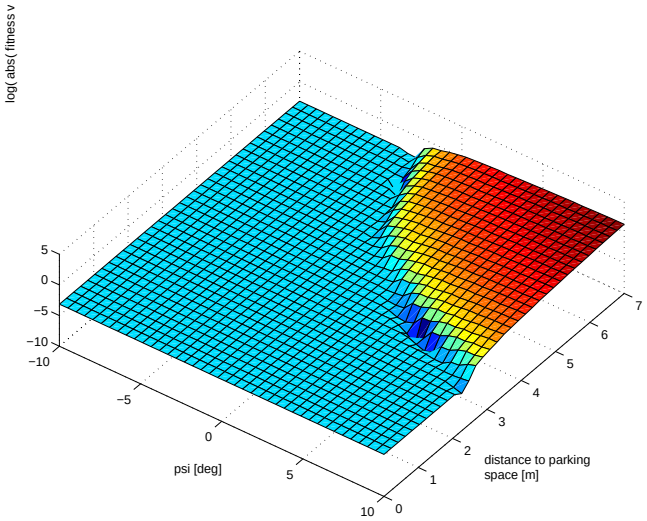

Fig.16: Logarithmic Representation of Absolute Fitness Values of Distance Criterion

The border to zero of both fitness functions have the same shape, like the diagrams in Fig.15 and Fig.16 illustrate. The surfaces show the function of angle psi and distance to parking space in the logarithmic representation of the absolute fitness value. This view on the values show, that the border of crossover from positive to negative fitness values is the same for both fitness functions.

LENGTH AND DISTANCE TO PARKING SPACE - This experiment varies the length of the parking space and the distance to the parking space. The range of the length, shown on the axis to the right, is between 5 and 10 m. The distance to the parking space, is shown on the axis to the left and ranges from 0 and 7 m. The remaining variables are kept constant, width of parking space was set to 2.5 m, the right gap beside the vehicle was set to 0.7 m and the angle psi was set to 0 deg.

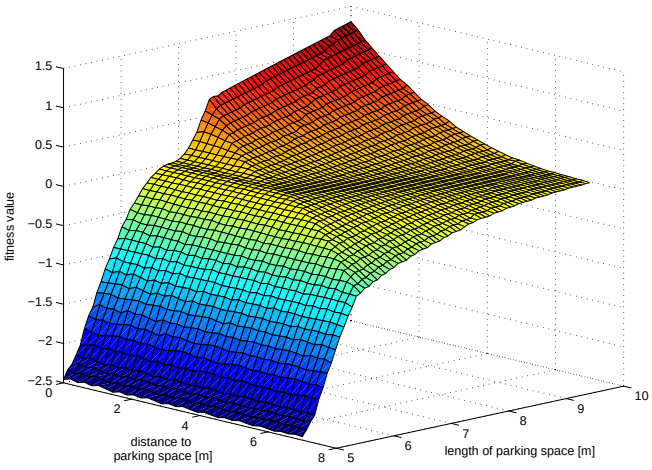

Fig.17: Fitness Values of Area Criterion as Function of Distance to and Length of Parking Space

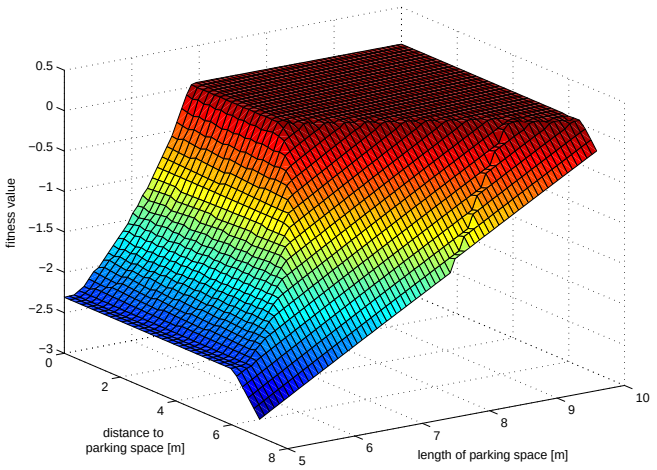

Fig.18: Fitness Values of Distance Criterion as Function of Distance to and Length of Parking Space

Like in the preceding example, the surfaces of the area criterion and distance criterion are different. The distance criterion surface (Fig. 18) has a flat plateau for low distances to the parking space and longer parking spaces. In contrast the area criterion shows for that domain a sloping characteristic (Fig. 17).

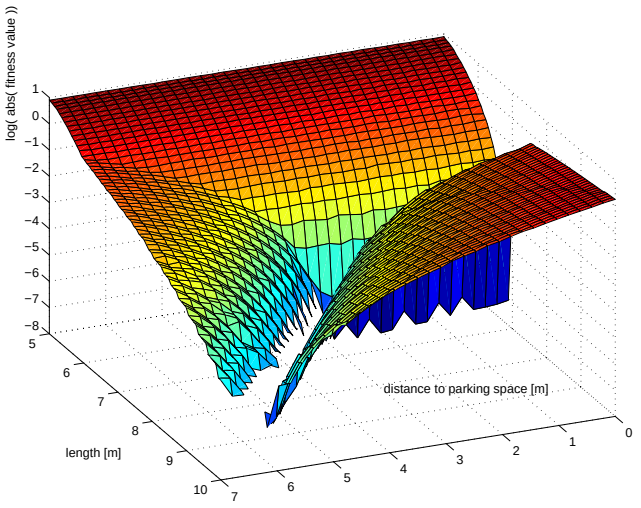

Fig.19: Logarithmic Representation of Absolute Fitness Values of Area Criterion

The surface of the fitness values converted into their logarithmic and absolute representation (Fig. 19, and Fig. 20) shows, that both fitness strategies have the same crossover from positive to negative fitness values, like in the example before.

CONCLUSION

The illustrations show, that both fitness functions have a sloping characteristic for collision maneuvers, where their fitness value is less than zero. The slope of the distance criterion function is more steep in that domain, compared

to the area criterion function. Furthermore, the distance criterion function shows an edge at the crossing to negative values, where the area criterion function has a smooth transition for fitness values around the zero. Nevertheless, both fitness functions show the same crossover characteristic from positive to negative values, like the logarithmic absolute representation illustrates. From that follows, that both types of function do identify the same scenarios where a collision happens.

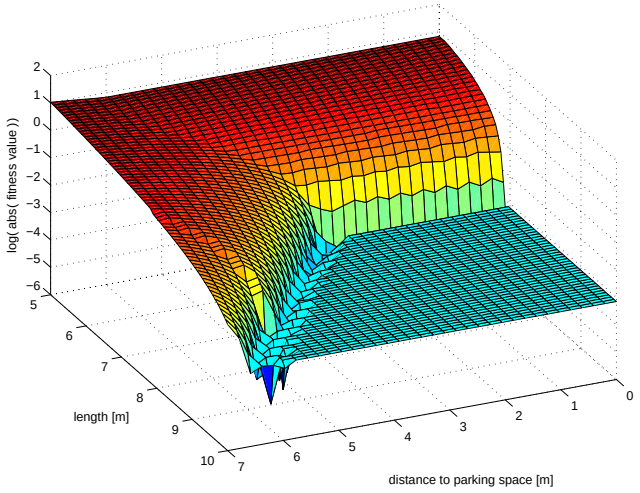

Fig.20: Logarithmic Representation of Absolute Fitness Values of Distance Criterion

The area criterion function provides a sloping characteristic for maneuvers with positive fitness values, in contrast to the distance criterion function, which returns constant fitness values for scenarios without clash and does not differentiate between good and better maneuvers. The consequence is, that the area criterion function can direct the search better towards collision maneuvers, than the distance criterion function. This supports a faster convergence for the optimization of the test cases to find faulty parking situations.

ACKNOWLEDGMENTS

The work described has been performed within the SysTest project. The SysTest project is funded by European Community under the fifth framework program (GROWTH, project reference G1RD-CT-2002-00683).

REFERENCES

1. Grochtmann, M., Grimm, K.: Classification-Trees for Partition Testing. Software Testing, Verification & Reliability, vol. 3, no. 2, pp. 63-82 (1993).
2. Wegener, J., Grochtmann, M.: Verifying Timing Constraints of Real-Time Systems by Means of Evolutionary Testing. Real-Time Systems, vol. 15, no. 3, pp. 275-298 (1998).
3. Tracey, N., Clark, J., Mander, K.: The Way Forward for Unifying Dynamic Test Case Generation: The Optimisation-Based Approach. Proceedings of the IFIP International Workshop on Dependable Computing and Its Applications, South Africa, pp. 169-180 (1998).
4. Schultz, A., Grefenstette, J., Jong, K.: Test and Evaluation by Genetic Algorithms. IEEE Expert, vol. 8, no. 5, pp. 9–14 (1993).
5. Jones, B., Sthamer, H., Eyres, D.: Automatic Structural Testing Using Genetic Algorithms. Software Engineering Journal, vol. 11, no. 5, pp. 299-306 (1996).
6. Pargas, R., Harrold, M., Peck, R.: Test-Data Generation Using Genetic Algorithms. Software Testing, Verification & Reliability, vol. 9, no. 4, pp. 263-282 (1999).
7. Michael, C., McGraw, G., Schatz, M.: Generating Software Test Data by Evolution. IEEE Transactions on Software Engineering, vol. 27, no. 12, pp. 1085-1110 (2001).
8. Tracey, N., Clark, J., Mander, K., McDermid, J.: An Automated Framework for Structural Test-Data Generation. Proceedings of the 13th IEEE Conference on Automated Software Engineering, Hawaii, USA (1998).
9. Pohlheim, H.: Genetic and Evolutionary Algorithm Toolbox for Use with Matlab - Documentation. http://www.geatbx.com/
10. Buehler, O., Wegener, J.: Evolutionary Functional Testing of an Automated Parking System. Proceedings of the International Conference on Computer, Communication and Control Technologies (CCCT '03) and the 9th. International Conference on Information Systems Analysis and Synthesis (ISAS '03), Florida, USA (2003).