Vision-Based Autonomous Navigation System Using ANN and FSM Control

Daniel Sales, Patrick Shinzato, Gustavo Pessin, Denis Wolf and Fernando Osório
Mobile Robotics Laboratory
University of São Paulo - USP
São Carlos, Brazil
dsales@icmc.usp.br, shinzato@icmc.usp.br, pessin@gmail.com, denis@icmc.usp.br, fosorio@gmail.com

***Abstract*— Autonomous mobile robot navigation is a very relevant problem in robotics research. This paper proposes a vision-based autonomous navigation system using artificial neural networks (ANN) and finite state machines (FSM). In the first step, ANNs are used to process the image frames taken from the robot´s camera, classifying the space, resulting in navigable or non-navigable areas (image road segmentation). Then, the ANN output is processed and used by a FSM, which identifies the robot´s current state, and define which action the robot should take according to the processed image frame. Different experiments were performed in order to validate and evaluate this approach, using a small mobile robot with integrated camera, in a structured indoor environment. The integration of ANN vision-based algorithms and robot´s action control based on a FSM, as proposed in this paper, demonstrated to be a promising approach to autonomous mobile robot navigation.**

Keywords-Mobile Robotics; Autonomous Navigation; Visual Navigation, ANN, FSM

I. INTRODUCTION

The application of Artificial Intelligence techniques to Autonomous Mobile Robots and Intelligent Vehicles have an important role in the international scientific robotics community [9][13][14]. One of the most desirable features in a mobile robot is the autonomous navigation capability. There are many important and well known works in this domain, as for example the Darpa Challenge (2004 and 2005 Grand Challenges at desert and 2007 Urban Challenge) [7][8] and the annual ELROB initiative [15][16], two of the most visible projects in this field of research.

Autonomous mobile robots usually execute three main tasks: localization, mapping, and navigation [17]. The localization task is related to estimating the robot's position in a known environment, using its sensors data. Mapping is responsible for creating a model to represent the environment based on robot's localization and sensors data. Navigation is the robot's capability to obtain information about the environment through its sensors, process it, and act, moving safely through this environment. In order to develop an Intelligent Autonomous Vehicle, capable of navigating into structured environments composed by roads and streets, one can assume that the robot already knows its approximate localization (e.g. using a GPS), the environment map and the path to be followed (origin/destination). Navigation in this environment consists basically to follow a well defined path, considering the road/street borders.

In this paper we focus on the navigation task, following a path defined by a road (road following task) which has a navigable area (inside the road) and non-navigable area (outside the road borders). Our main goal is to reproduce the control of a vehicle navigating into a road, using a vision-based system, and following the path defined by the road in a small scale (using a small robot). This small vehicle should also be able to decide when/how to proceed in order to turn left or right, even when the visual information about the road is out of its field of view.

Our vision-based navigation approach uses a group of Artificial Neural Networks (ANNs) combined with the implementation of a Finite State Machine (FSM) to autonomously control a robot. The mobile robot platform used is a Surveyor SRV-1 robot with an integrated camera and wireless connection (Wi-Fi). The experiments were conducted in an indoor environment reproducing a typical road following navigation task.

The next topics of this paper are organized as follows: Section II presents a review of some related works; Section III presents the techniques and features used to identify the navigable region in the image, identify the current state and act, moving the robot through the environment; Section IV shows the experimental results obtained from tests in the real environment; Section V presents the conclusion and future works

II. RELATED WORKS

Many different approaches were developed for navigation, using different types of sensors (e.g. laser, sonar, GPS, IMU, compass), individually or grouped [9][17][18]. One of the currently most studied approaches is the vision-based navigation methods. These methods adopt cameras as the main sensor. Cameras proved to be very suitable sensors for route following and obstacle avoidance because of its light weight and low energy consumption [1]. Moreover, an image can give many types of different information about the environment at the same time, without requiring to work

978-0-7695-4231-7/10 $26.00 © 2010 IEEE
DOI 10.1109/LARS.2010.26

fusioning information from several different sensors. It is also possible to reduce costs using a camera instead of other sensors [2].

In the implementation of route following and navigation systems, for structured or semi-structured environments, it is usual to implement vision-based approaches [3][7][9][10] [11]. These systems classify the image, segmenting the road region, and identifying the navigable area in front of the vehicle. The output of these systems is an image with the road surface segmented, indicating a safe zone for navigation.

Although the previously mentioned techniques provide good results, the field of view of commercial cameras is very restricted, and many solutions require the fusion of data from laser sensors (e.g. Sick Lidars, IBEO, Velodyne), radar sensors, and/or special vision systems (e.g. omnidirectional cameras) [7][8] [9][18]. These solutions are very expensive, where in our proposed approach we would like to use only one single usual/commercial camera.

In order to validate our autonomous navigation system, a simple reactive image-based system was not adequate, since, as described above, the camera field of view is very restricted, and the immediate reaction to the information provided by the vision system is not enough to guarantee the correct mobile robot control. A more robust control system should be implemented.

In Robotics, Finite State Machine (FSM) [19] based approaches are often used [20][21], as for example, the "Situated Automata" and the "Reactive Deliberation Architecture". FSMs are useful because the system can be easily described as a sequence of states (context changes), taking into account: ***inputs*** (sensors) that allows changing from one ***state*** (situation) to another one, and also defining for each state a specific ***action*** (motor action) associated to it. So for each state and state change, the robot is able to react properly. We chose to implement our control system based on this main idea that the mobile robot control system can be described by a FSM, using as inputs the route detection information obtained from images..

III. METHODOLOGY

In order to safely navigate through the environment, it was needed a method to identify the navigable areas. In the work proposed by Shinzato [3], a set of ANNs was used to identify the navigable region, and its performance was evaluated, obtaining good results. The system combines the output of four Multi-Layer Perceptron (MLP) networks, adopting a group of networks [12] to identify the navigable region. This combination's result is a visual navigability map which can be used as input for the FSM based navigation control method.

As done in [3], a block-based classification method was used. It consists on dividing the image in blocks of pixels evaluated as a single unit. A value is generated to represent this group, and this value can be the average of the RGB (Red-Green-Blue color channels), HSV (Hue-Saturation-Value color attributes), entropy, and other features from the set of pixels represented in this block. In the image slicing step, a frame with resolution (M x N) pixels is sliced in groups with (K x K) pixels, as shown on Fig 1.

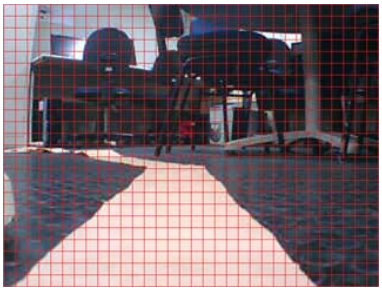


Fig. 1. Frame slicing: 320x240 image sliced into blocks of 10x10 pixels

The captured image is represented by a matrix I of (M x N) size and the processed image is represented by a matrix G of (K x K) size. The element $I(m,n)$ corresponds to the pixel in row m and column n of image, where ($0 \leq m < M$) and ($0 \leq n < N$). Therefore, group $G(i, j)$ contains all the pixels $I(m, n)$ in such a way that (($i * K$) $\leq m <$ (($i * K$) + K)) and (($j * K$) $\leq n <$ (($j * K$) + K)). This strategy has been used to reduce the amount of data, allowing faster processing.

After calculating the attributes of all image's blocks, they are ready to be classified by an ANN. Four ANNs are used, each of them classifies the block as navigable or non-navigable, receiving one block's attributes as input, and giving 0.0 or 1.0 as output (0 for non-navigable and 1 for navigable).

In this work, the used ANNs have one hidden layer with five neurons, the output layer with one neuron and the input layer with four or five neurons (Fig 2), according to the features used as inputs.

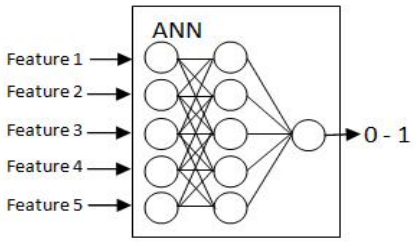


Fig. 2. ANN topology. Image attributes as input, and classification as output: 0 for non-navigable and 1 for navigable

The main difference between the four ANNs used is the attribute set used as input. These attribute sets (see Table 1) are calculated during the segmentation of the image in blocks. The attribute choose was made based on results from work [3]. More information about these attributes can be found on [4],[5] and [6].

After obtaining the four outputs for each block, the average value between these four values is calculated, in order to

compose the final value for each position of the navigability matrix. Figure 3 shows the structure of the classifier which combines the outputs of the ANNs.

TABLE I. INPUT ATTRIBUTES FOR EACH USED ANN

ANN	Input Attributes
ANN1	R average, B average, H average, V entropy and HSV energy
ANN2	R average, H average, H entropy and V entropy
ANN3	B average, S entropy, V entropy, S energy and HSV entropy
ANN4	B average, V entropy, S energy, S variance and RGB entropy

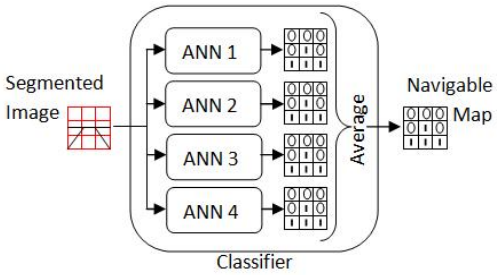


Fig. 3. Classifier Structure. Value in navigability map is the average between the outputs of the ensemble of four ANNs

As the final output values vary between 0.0 and 1.0, several levels of certainty could be set in the navigability matrix, so in order to avoid this problem, the values were approximated to:

- 0.0, if output $\leq$ 0.3, representing non-navigable block;
- 1.0, if output $\geq$ 0.7, representing navigable block;
- 0.5 if 0.3 < output < 0.7, representing doubt about the navigability of the block;

The training phase of the ANNs is done with the first frame captured. This training is done through an interactive interface where the user must select the navigable and non-navigable blocks, thus calibrating the system to current route visual conditions. Then, the training's result is shown and the user can determine if a new training is needed or if the autonomous navigation can be started.

Once the ANN training process is complete, the system is able to start the navigation step. The navigation is based on a FSM control, so is possible to say that the system is not purely reactive, because it does some kind of planning based on memory of last state detected (situation context).

This type of planning is needed because there are some special situations, for example 90º turns, where the system detects the turn before the point where the robot really needs to turn. At the turning point the route will not be visible anymore. To solve this problem, it is necessary to keep the information about the side of the turn detected and then, when the route is not visible anymore, make the robot turn to that side.

This context-dependent task can be easily implemented using a simple FSM. If we adopt a route map, the FSM can be also used to decide if the robot should turn to the left, to the right, or go straight ahead when arriving to an intersection of roads. The FSM can be easily extended in order to take care of several different situations.

The first step of the FSM control stage is to extract the appropriate information from the image containing the segmented route (navigable area) and use it as input. This input is evaluated comparing the data with the default values which define each state.

One peculiarity of structured environments is to show some kind of pattern in the navigable area. In other words, different roads may have some similarities when captured by a camera, as shown on Fig 4, where four different straight roads pictures have the same shape.

Fig. 4. Photos of four different straight roads. When processed, all of them will result in a triangle as the navigable area

The navigability matrix for most straight route segments is fulfilled in a similar way, defining an easily recognizable shape. As defined in straight paths, each route configuration has a specific pattern associated, so this feature is the basis of FSM's state detection. In order to identify the current state, it was needed to develop a way to evaluate the navigability matrix, identifying a specific route configuration. This is done analyzing specific areas of the navigability matrix which contain the wanted information about the route configurations as shown on Fig 5. The A0 zone is the area above the horizon line in the original frame, and does not affect the result, so it is not processed. A1, A2 and A3 are the areas of interest, which can help us to determine the route shape and input parameters for the FSM.

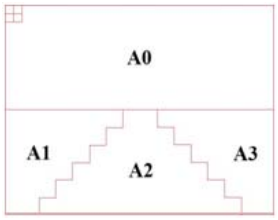


Fig. 5. Areas of interest in the navigability matrix.

As previously shown, the matrix is fulfilled with 0 (zeros) for non-navigable blocks, 0.5 for uncertain blocks or 1 (one) for navigable blocks, so it is easy to deduce that in a straight path for example, most elements inside A2 are equal to 1's and elements inside A1 and A3 are 0's. Likewise, in a "left turn" state, A1 has some elements with value 1, A2 has all elements with value 1, and A3 has most elements with value

0. Thus, the average between all the values of each area is calculated, and every state has defined his own typical value set for A1, A2 and A3. The states and their associated actions we implemented, in order to execute the experiments described in the next section, are shown on Table 2.

TABLE II. FSM STATES AND ACTIONS

State	Related Action
Straight route	Go forward
90º left turn	Keep left turn in memory and go forward
90º right turn	Keep right turn in memory and go forward
Left turn	Smooth turn to the left
Right turn	Smooth turn to the right
End of Route	If any side turn is in memory pivot 90º to that side, otherwise smoothly spin to left

After detecting the state, a command is sent to the robot, in order to move it through the environment. Then, a new image is captured, and all process is repeated. The full navigation system's diagram is shown on Fig 6.

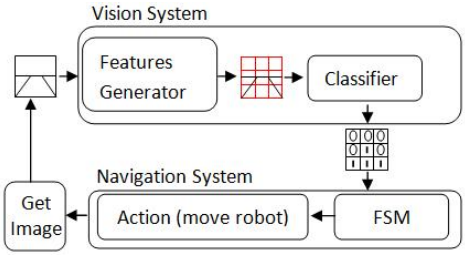


Fig. 6. Proposed method flowchart.

It is important to observe that: (*i*) the vision system can be trained to recognize different types of road images, like asphalt, different pavements, natural soil roads, or even artificial roads as the one used in our experiments; (*ii*) the FSM can be structured in order to recognize different situations and to execute different types of actions, so we can easily extent the set of mobile robot behaviors. The adaptability of our system to different situations and tasks, demonstrates the flexibility of the proposed method.

IV. EXPERIMENTS AND RESULTS

Experiments were accomplished using a mobile robot, Surveyor SRV-1Q [22] (Fig. 7). This is a small robot developed for academic purposes, equipped with a 500 MHz MIPS processor, 32Mb RAM memory, digital camera with resolution ranging from 160x128 to 1280x1024 pixels, two laser pointers and wireless 802.11b/g. It has also DHCP/HTTP server running on-board. By default, the connection is established in Ad-Hoc mode, and the robot is controlled by sending/receiving packets through sockets.

A programming interface (API) in "C" was not provided by the robot's manufacturer, so specific high-level functions, necessary to our implementation and to allow the integration to the other software modules, were not available. In order to simplify the programming process and to make it less error-prone, a library with high-level functions was developed in C/C++[1].

Fig. 7. Surveyor SRV-1Q tracked robot.

Basically, the set of classes organizes sending and receiving messages through TCP/IP sockets. This way, high-level commands can be easily written. In order to create a "platform-free" code, GNU/Linux's and Window's default libraries (<sys/socket.h> and <winsock.h> respectively) were not used; instead of using those libraries we choose to adopt SDL_Net library [23] which is free, open source and portable. SDL is very commonly used in multimedia applica-tions, such as network games, and many operating systems are supported. The ANNs were implemented using FANN [24], a free open source neural network library which implements multilayer artificial neural networks in C. Image processing routines were implemented in OpenCV [25], a free open source library for real time computer vision.

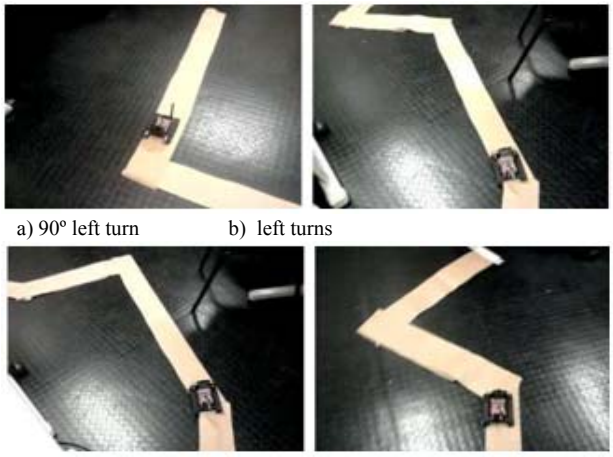
The frame size used is 320 x 240 and each block has 10 x 10 pixels size, resulting in a 24 lines x 32 columns naviga-bility matrix. The frame rate varies, because a new frame is taken only after moving the robot, as shown previously.

The control program developed executes the steps presented in the previous section (ANN and FSM) in a Linux platform. All the main processing is done by a remote computer, which receives the images from the robot, runs the program and sends back commands to the robot. It communicates via Wi-Fi with the robot receiving the captured frame, and sending the commands to control the robot's motors.

The first step is to capture the first frame, and open an interface where the user may define which are the navigable and non-navigable blocks. The training is done, and the user can choose if a new training will be required or if the autonomous navigation can be started. After this, the robot starts to move autonomously, whitout any user intervention. All captured frames are saved at the computer's disk, and navigability matrix for each frame too. Each frame is mapped into a three-color image frame: black for non-navigable points, white for navigable and grey for uncertain about the navigability.

[1] Source code available at:
[http://www.lrm.icmc.usp.br/wiki/index.php/SRV-1Q]

As SRV-1Q is as small indoor robot, the experiments were realized in a structured indoor environment, made with similarities to outdoor structured environments in order to evaluate the technique and validate its results. Fig 8 shows some scenes of the created environment with the robot navigating through them. Many situations commonly found in real outdoor road following tasks were simulated, as "V turns", smooth turns, 90º turns and straight paths.



a) 90º left turn b) left turns

c) 90º left turn after smooth turn d) Right turn with more than 90º

Fig. 8. Some points of the path which simulates situations of outdoor structured road following environments

The robot performed well and as expected on all of presented situtations. Some videos demonstrating the robot's navigation following a route based on the ANN and FSM control are available at youtube, on LRM's channel (http://www.youtube.com/watch?v=K1c-6r_-CxY).

In a single straight route, the robot's behaviour is to navigate forward from the beginning to the end of route. Then, the robot turn to left every time it finds the road end, turning to left until it finds the the route again and aligns to it, starting to follow the route again.

This also happens when, for any reason, any part of the road is not detected. For turns with less or more than 90º, the robot turns slowly until aligning with the road. This way the robot is able to follow a road, given preference to turning to left, if no previous context information (memory) was provided by the FSM.

Fig 9, Fig 10 and Fig 11 show the comparison between the original frame taken from robot's camera and the frame generated representing the navigability map of three different states. In all of them the detection of the road was accurate.

Fig 12 and Fig 13 show some error observed during navigation. Due to the limitations of the vision system initial training, some objects were wrongly classified as navigable points (see Fig 12) and could interfere on state detection if they were inside any of the control areas observed (areas of interest as indicated in Fig. 5).

Fig 9 – Comparison between a straight way frame and the navigability map generated from it

Fig 10 – Comparison between a 90 º left turn frame and the navigability map generated from it

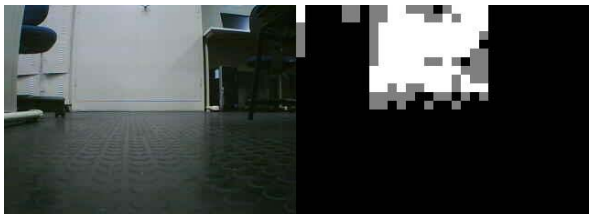


Fig 11 - End of route state

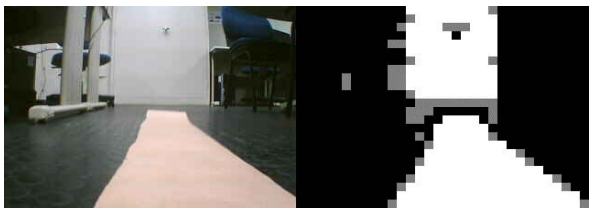


Fig 12 - Cabinet wrongly classified as navigable

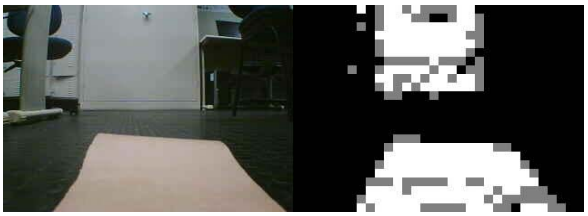


Fig 13 - Noisy navigability map

Another problem can occur when the local illumination conditions change. Points in the route with different light incidence were not classified accurately, as shown on Fig 13. The number of uncertain image blocks, represented by gray blocks, or even the number of non-navigable blocks in this scene is higher. Once the uncertain block values does not affect the areas of interest average (A1,A2,A3 – representing the route direction), sometimes is possible to detect a wrong state input, when the ammount of these points is too large. Consequently, the FSM is not very robust to noisy inputs.

Despite these problems, the system has been accurate in almost all test cases. In normal conditions the mobile robot achieved 100% of correct task executions, only when the local illumination was not adequate or when the image proceesing ANN training was not properly executed, the robot failed in the road following task.

V. Conclusion and Future Works

The implemented method obtained very good results, showing that vision-based navigation strategy integrated to a FSM control is a very convenient approach for mobile robot navigation.

Considering the fact that the vision system can be re-trained to recognize different types of road images, we expect that this approach could be quickly adapted to be used by an outdoor vehicle (as suggested also by the experiments presented in [3]). The FSM can also be quickly adapted to recognize different types of situations and execute different types of actions, so we are sure to be able to easily extent the capabilities of this initial system. The proposed method demonstrated to be flexible in order to be adapted to different situations and applications.

On the other hand, there are several ways to improve the work presented in this paper. In order to avoid the method´s imprecision issues and simplify the system application even in more complex environments, more robust techniques should be investigated: automatically detection of environment illumination conditions changing with self re-trainable ANNs; creation of more robust FSMs which may possibly deal with noisy inputs (improve the FSM control by another method, based on machine learning for example); introduction of more complex behaviors in the robot control, as for example, obstacle detection and avoidance. Our research group is currently working on these issues.

Acknowledgment

The authors acknowledge the support granted by CNPq and FAPESP to the INCT-SEC (National Institute of Science and Technology - Critical Embedded Systems - Brazil), processes 573963/2008-9 and 08/57870-9. Also acknowledge CAPES and CNPq for their financial support of this research (doctoral grant).

References

[1] Zingg, S., Scaramuzza, D., Weiss, S., and Siegwart, R. MAV Navigation through Indoor Corridors Using Optical Flow , *IEEE International Conference on Robotics and Automation (ICRA 2010)*, Anchorage, Alaska, May, 2010.

[2] Scaramuzza, D., Siegwart, R. Appearance Guided Monocular Omnidirectional Visual Odometry for Outdoor Ground Vehicles. *IEEE Transactions on Robotics*, vol. 24, issue 5, October 2008.

[3] Shinzato, P. Y, Wolf, D. F. Features Image Analysis for Road Following Algorithm Using Neural Networks. September, 2010. *Accepted to 7th IFAC Symposium on Intelligent Autonomous Vehicles 2010 (IAV)*. Lecce, Italy

[4] Shannon, C., E., A mathematical theory of communication, Bell System Technical Journal, vol27, 1948.

[5] Joblove, G., H., Greenberg, D., Color spaces for computer graphics. SIGGRAPH, Comput. Graph., v. 12, n.3, p.20-25, 1978.

[6] Reiter, C., With j: image processing 2: color spaces. SIGAPL APL Quote Quad, v.34, n. 3, p.3-12, 2004.

[7] Thrun, S. et al. (2006) "Stanley: The Robot that Won the DARPA Grand Challenge," Journal of Field Robotics, Vol. 23, No. 9, June 2006, p.661-692. http://robots.stanford.edu/papers.html (Visited 08/02/2009).

[8] Urmson, Chris et al. (2008). “Autonomous driving in urban environments: Boss and the Urban Challenge”. In: Journal of Field Robotics. Vol. 25 , Issue 8 (August 2008). Special Issue on the 2007 DARPA Urban Challenge, Part I. Pages 425-466.

[9] Buehler, Martin; Iagnemma, Karl; Singh, Sanjiv (Editors). The 2005 DARPA Grand Challenge: The Great Robot Race (Springer Tracts in Advanced Robotics). Springer; 1st. edition (October, 2007).

[10] Nefian, A.V.; Bradski, G.R. (2006) “Detection of Drivable Corridors for Off-Road Autonomous Navigation”. ICIP-06: Proceedings of the IEEE International Conference on Image Processing. pp. 3025-3028.

[11] J.M. Álvarez, A. M. López, and R. Baldrich. (2008) “Illuminant Invariant Model-Based Road Segmentation”. IEEE Intelligent Vehicles Symposium, Eindhoven, Netherlands, June 2008. http://www.cvc.uab.es/adas/index.php?section=publications

[12] Bishop, C.M. (1995) Neural Networks for Pattern Recognition, Oxford: Oxford University Press.

[13] Bishop, R.; (2000) "Intelligent vehicle applications worldwide". IEEE INTELLIGENT SYSTEMS - Intelligent Systems and their Applications, Volume: 15, Issue: 1. pp.78-81.

[14] Bishop, Richard. (2000). "A Survey of Intelligent Vehicle Applications Worldwide". Proceedings of the IEEE intelligent Vehicles Symposium 2000. pp.25-30.

[15] Kuhnert, K.-D. (2008). "Software architecture of the Autonomous Mobile Outdoor Robot AMOR". Proceedings of the IEEE Intelligent Vehicles Symposium, 2008. pp. 889-894.

[16] Schilling, Klaus. (2008) “Assistance Systems for the Control of Rovers”. SICE Annual Conference, Tokyo, Oct. 2008.

[17] Wolf, Denis F.; Osório, Fernando S.; Simões, Eduardo; Trindade Jr., Onofre. Robótica Inteligente: Da Simulação às Aplicações no Mundo Real. [Tutorial] In: André Ponce de Leon F. de Carvalho; Tomasz Kowaltowski. (Org.). JAI: Jornada de Atualização em Informática da SBC. Rio de Janeiro: SBC - Editora da PUC. RJ, 2009, v. 1, p. 279-330.

[18] Goebl, M.; Althoff, M.; Buss, M.; Farber, G.; Hecker, F.; Heissing, B.; Kraus, S.; Nagel, R.; Leon, F.P.; Rattei, F.; Russ, M.; Schweitzer, M.; Thuy, M.; Cheng Wang; Wuensche, H.J.; (2008) "Design and capabilities of the Munich Cognitive Automobile". IEEE Intelligent Vehicles Symposium, 2008. Page(s): 1101 – 1107

[19] Hopcroft, J.E., Ullman, J.D. (1979) “Introduction to Automata Theory, Languages and Computation”. Addison -Wesley, 1979.

[20] Medeiros, Adelardo A. D.. “A survey of control architectures for autonomous mobile robots”. Journal of the Brazilian Computer Society - JBCS. 1998, vol.4, n.3.

[21] Sahota, Michael K. “Reactive Deliberation: An Architecture for Real-Time Intelligent Control in Dynamic Environments”. Proceedings of the AAAI-94, p. 1303–1308.

[22] Surveyor Corporation, http://www.surveyor.com/SRV_info.html, Accessed in May, 2010.

[23] Simple DirectMedia Layer, http://www.libsdl.org/projects/SDL_net, Accessed in May, 2010.

[24] Fast Artificial Neural Network Library, http://leenissen.dk/fann/, Accessed in May, 2010.

[25] Open Source Computer Vision, http://opencv.willowgarage.com/wiki, Accessed in May, 2010.