FSM-based Visual Navigation for Autonomous Vehicles

Daniel Oliva Sales[1], Leandro Carlos Fernandes[1], Fernando Santos Osório[1] and Denis Fernando Wolf[1]

Abstract — **In this paper, we present a vision-based navigation system for autonomous vehicles in structured urban environments. This system uses a standard RGB camera as the main sensor. The proposed approach is composed by an association of Finite State Machines (FSM) with Artificial Neural Networks (ANN). First, we identify the navigable areas after processing the input frames. Then, an ANN is trained to recognize patterns on the generated navigability maps. Each pattern is associated to a specific state, related to a unique environment structural feature. A topological-like map is used to represent the environment, so any path can be described as a sequence of states. For example, straight path, right and left turns and intersections. The experiments were performed with an autonomous vehicle in a real urban environment in order to validate and evaluate this approach. The proposed system demonstrated to be a promising approach to autonomous vehicles navigation.**

I. Introduction

The development of robust autonomous intelligent systems for robotic applications is a very important research topic. Several applications are related to robotics, from industry to military tasks.

The autonomous driving capability is one of the most desirable features for a mobile robot. Researches related to this feature are being developed since the 80´s, and groups such as NavLab have been presenting relevant results on autonomous vehicles navigation.

Nowadays there are many relevant and known researches on autonomous robotics being developed worldwide. Some of them are powered by government initiatives as for example the Darpa Grand Challenge [7][8][9]. The first two editions (2004 and 2005) were held in desert, and 2007 edition in an urban environment. In Brazil, the development of autonomous vehicles is an important research challenge among the main working groups (WG2) of the Brazilian National Institute of Science and Technology on Embedded Critical Systems (INCT-SEC).

Autonomous mobile robots usually perform three main tasks: localization, mapping and navigation [17]. Mapping is the creation of an environment model using the sensorial data, representing its environment structure. Localization task must occur simultaneously to navigation control. It consists in estimating the robot´s position in a previously known environment, using its sensorial data. The Navigation task is therefore the ability to obtain enough information about the environment, process it, and act, moving safely through the navigable area.

*Research supported by FAPESP.

The authors are with the Mobile Robotics Lab at ICMC - University of São Paulo, São Carlos, CEP 13566-590, Brasil (e-mail: dsales, lnd, fosorio, denis @ icmc.usp.br).

In order to develop an intelligent system able to navigate through environments composed by streets and highways, it is desirable to know the robot´s approximate position, the environment map and the path to be followed. So, navigation in this environment lies in following a well-defined path, considering the navigable areas.

This work focuses on this navigation task, describing the development of a Vision-Based Topological Navigation System, able to recognize the navigable area of an urban environment (streets) processing image frames and classifying them into states which represent the current robot context, allowing the robot to autonomously drive through this environment and reach a desired destination.

The adopted navigation approach does not require a very detailed environment map (metric map), only a graph that represents the main elements, in a simpler path representation. Furthermore, accurate pose estimation is not necessary. The approximate robot´s position is enough to navigate. So, the main objective is to detect the current node in a Topological map, being useful to autonomously decide when and how go straight, turn left or right, even when more than one possibility is detected simultaneously (at an intersection, for example).

The developed system uses Artificial Neural Networks (ANN) [19] in two steps: to classify the frame obtained from camera (resulting in a navigability map) and second to detect patterns on these navigability maps (representing the current context). In this second step, a Finite State Machine (FSM) is used to represent the state sequence for any path at the environment. The ANN is trained to recognize all possible states, so a FSM generator can convert any chosen path into a sequence of these known states.

The motion control is based on hierarchical/hybrid control approach. This way, the navigation system combines the high-level deliberative control (path planning) with different reactive behaviors, allowing a safe motion. This FSM-based approach was already successfully applied in previous authors' works for indoor applications [4][25][16] with different sets of sensors and states, showing the feasibility of this implementation and motivating studies concerning the application of this technique in urban environments.

The main objective of this work was obtaining and processing enough information for state (context) detection, allowing a high-level path planning and also safe motion using reactive control for autonomous vehicles.

The next topics of this paper are organized as follows: Section 2 presents some previous related work; Section 3 presents the Topological Navigation System overview, Section 4 presents experiments and results and Section 5 presents the conclusion and future work.

II. Related Work

Several navigation approaches have been used for navigation, using many different sensors (for example laser, sonar, GPS, IMU, compass) solely or combined [9][17][18]. One of the most used approaches is the vision-based navigation [20]. This method uses monocular video cameras as the main sensor. Cameras are very suitable for navigation and obstacle avoiding tasks due to its low weight and energy consumption [1]. Furthermore, one single image can provide different types of information about the environment simultaneously. It is also possible to reduce costs by using cameras rather than other types of sensors [2].

Vision-based navigation approaches are already usual in navigation systems for structured or semi-structured environments [3][10][11]. These systems classify the image, with track segmentation for safe navigable area identification, resulting in reactive models for navigation control. Works such as ALVINN [13] and RALPH [14] were some of the first to apply neural networks for this reactive control in outdoor environments.

In [3], Shinzato developed a neural classifier composed by an ANN ensemble, able to detect and to segment the navigable areas of the road through image features analysis. Later, this classifier was adopted by Souza in [15] for a Template-Matching based reactive control.

Purely reactive models are not totally adequate for our autonomous navigation system development, since immediate reaction to sensors data is not enough to guarantee a correct control in complex environments. A more robust system should be implemented, providing sequence and context information that are absent in purely reactive models.

In robotics, FSM-based approaches [5] are very often used. FSMs are useful because the system can be easily described as a sequence of states (context changes), considering the inputs (sensors) and specific actions for each state. This way, for each detected state the robot can assume a different behavior. This work focuses on this main idea, with possible paths described as FSMs in which current state is detected after processing sensors data.

The use of Machine Learning techniques such Artificial Neural Networks is a very interesting way to process input data, identifying and classifying the states and transitions to determine the best actions to be performed [6]. ANNs are tolerant to noise and imprecision on input data, and able to detect the states and transitions between these states. ANNs are also very efficient to generalize knowledge (adjusting the outputs to many inputs, even the ones not explicitly taught to the net). So, this technique is very useful for state detection through path features recognition.

The association of ANNs with FSMs is being developed and evolved since the 1990´s [21][22][23][24], and recent researches were focused on applying this technique to robotics. In [12], an autonomous car parking system was developed, using recurrent neural networks for FSM learning. The sensors data and current state were used as ANN inputs, so the system could detect when a context change was needed. This work inspired the development of our FSM-based approach.

In [20], a Vision-based autonomous navigation system was implemented for indoor environments using a simple FSM control. In that work, Sinzato´s classifier was used to generate the navigability matrices from input frames. Then, an algorithm was developed to analyze interest areas of these matrices in order to determine the current state of a FSM (robot context) to navigate through a set of turns and straights. The main idea of this FSM-Control was evolved, so in [4] and [25] an autonomous navigation system was developed with an ANN as the control unit. A LIDAR sensor was adopted as the main sensor, so an ANN was trained to recognize the "laser signatures" associated to every possible state of a FSM. Each state was related to a specific part of an indoor environment. This way, the FSM states could be learned by the ANN, and the paths represented by a sequence of known states. These works introduced the Topological Navigation approach proposed in this paper.

The navigation system developed in this paper combines the visual (low-cost) navigation with the neural learning of FSMs in Topological Navigation approach, considering the best features obtained with these previous results. The developed system components are described in next section.

III. System Overview

The proposed system is composed by three main modules: "Navigability Map Generation", "State detection" and "Reactive Control". The Navigability Map generator is responsible to convert a captured frame into a navigability map, used as input for our Navigation Control System. The State Detection module is responsible to detect the current state and filter oscillations, avoiding unexpected state transitions (considering the path plan). The reactive control determines an appropriate steering angle according to the navigable area detection and current state detected. Figure 1 shows the full system flowchart.

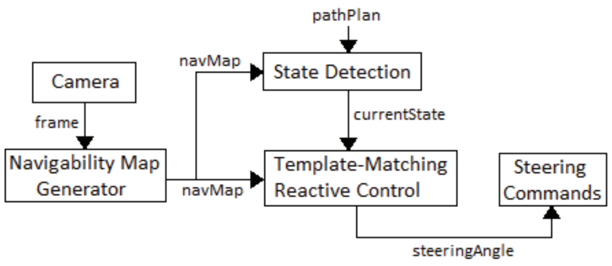


Figure 1 – Navigation System Flowchart

A. Navigability Map Generation

This step is performed using the neural classifier proposed by Shinzato in [3]. It is composed by an ANN ensemble with six ANNs; each one is responsible to classify the pixels into navigable (1) or non-navigable (0), based on different sets of features of the image such as RGB values, HSV values and entropy. So, the mean of these output values is the final classification value, ranging from 0 to 1.

The ANNs training is performed with a small set of representative frames. The supervisor must classify parts of these frames into navigable or non-navigable in a GUI, obtaining six trained ANNs.

The input frame has 320x240 pixels size, sliced in 10x10 block to produce a 32x24 navigability map. The neural classifier and the image features used for each ANN are fully described in [3]. Figure 2 shows some examples of processed frames and the resulting navigability maps. The segmented navigable area is used as input for the State detection and Reactive Control modules.

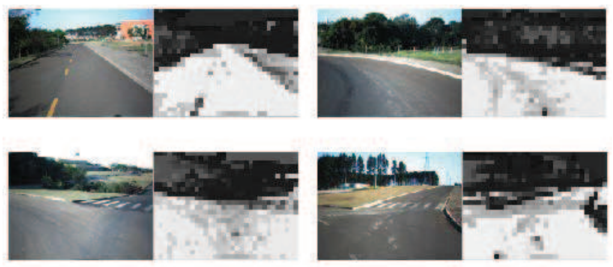


Figure 2 – Navigable area detection results on different situations (bright = navigable block)

B. *State Detection*

This module is responsible to recognize patterns on the input data, allowing the detection of environment features used to determine the current state (context). An ANN is used for this task.

The environment is mapped as a topological map in which each node is related to a specific part of the environment, described by its structural features such as straight path, turn or intersections. All possible states (environment features) are taught to the ANN, so possible paths on this environment can be described as a sequence of these learned states, being represented by FSMs. Figure 3 shows a part of an environment and its topological map.

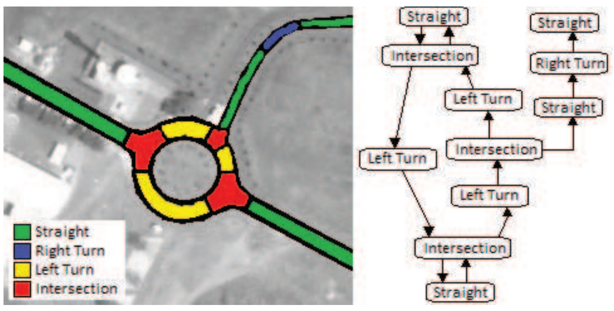


Figure 3 – Environment mapping example

The ANN input is the navigability matrix obtained at the previous step, and the output is the current state (environment feature) detected. Four different classes were created, each one related to a different track shape or condition (straight, left and right turns and intersection).

In order to begin the training process, a database must be generated collecting a set of frames for each possible situation. So, these frames may be processed by the first neural classifier to generate the navigability maps. As the learning process is supervised, a specialist must classify these navigability maps in one of the possible states before the final ANN training, creating a set of input/desired-output pairs. After finishing the ANN training, it should be able to recognize different possible paths at the environment.

Once the Topological Map of an environment is known, it is possible to establish a route between two points, manually or with a path planning algorithm. Every route can be seen as a sequence of steps (states), so it is trivial to generate a FSM (sub-graph) to represent a well-defined path.

For this system implementation, it is assumed that vehicle´s initial position is always known, as the topological map also. The current position is estimated based on current state detection, so it isn´t necessary to estimate robot´s exact position. Navigation and self-localization are performed together.

The desired path is also used as input to State Control unit, so the system can determine if a state transition is needed or if the vehicle still remains at the current state. The State Control unit also filters state detection oscillations, allowing a state transition only if the next expected state is observed by a certain amount of consecutive steps. If the observed state does not match the path plan, current state is kept, as shown on Figure 4.

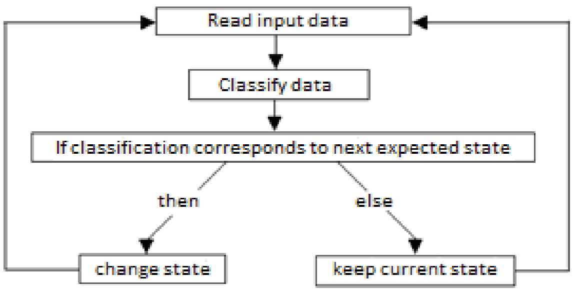


Figure 4 – State transition flowchart

This Topological Navigation approach allows the vehicle to follow its planned path and also know its approximate position, but does not control the motion "into" every situation (state). This task is performed by a reactive control unit, described in the next section.

C. *Template-Matching Based Reactive Control*

This module is responsible to react to road current conditions, determining the vehicle´s steering angle. As mentioned earlier, the motion control system must combine the deliberative control resulting from FSM-based topological navigation with reactive behaviors to guarantee a safe driving, avoiding leaving the track.

The implemented reactive control uses a template-matching approach for road track following, as proposed in [15]. The navigability maps are compared with five different templates, each one related to a specific steering angle for reaction, as shown on Figure 5. Each template receives a matching score, so the steering angle is defined after measuring the best score.

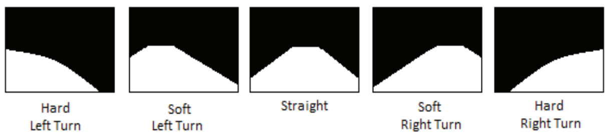


Figure 5 – Reactive Control Templates representation

The current state is also considered as input, so a bias is added to the score of the most suitable templates for each current condition. Straight path state adds a bias to straight and soft turns templates, avoiding abrupt movements; left and right turns adds a bias to its corresponding soft and hard turn templates; intersection state adds a bias to straight, hard and soft turns considering the path plan (left to keep in the roundabout, right to get out).

IV. Experiments and Results

The experiments were carried out in a real urban environment, using an autonomous vehicle equipped with a video camera as the only sensor. The environment was composed by straights, turns and intersections, so its properties could be represented with four states: "straight path", "left turn", "right turn", and "intersection". Some examples of input frames for these four states are shown on Figure 6.

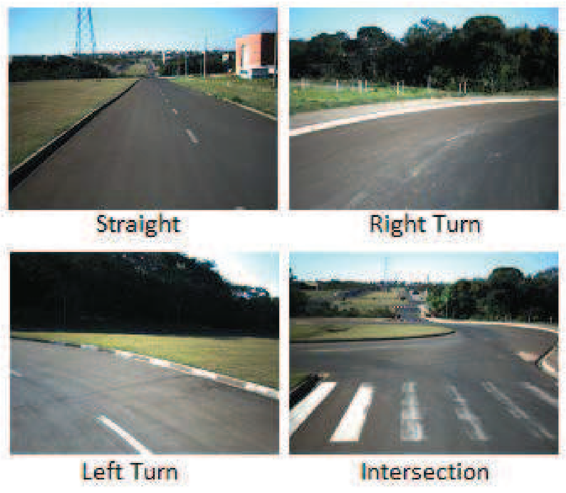


Figure 6 – Example of input frames for each implemented state

A. System Setup

The camera used in our tests was a PointGrey Bumblebee2 Stereo Cam [26]. We do not deal with depth information in this paper, so only left image is taken as perception system input. The camera was placed above the car as shown on Figure 7, so viewing angle was adjusted in order to allow immediate reaction to each frame/navigability map processed.

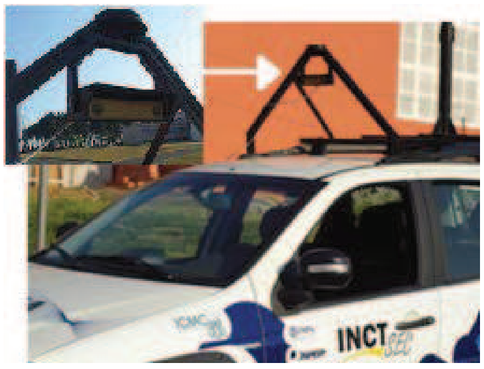


Figure 7 – Camera position

The control platform was implemented on Robot Operating System (ROS) [27], a framework which provides libraries and tools for robot applications development. Each module was implemented as a ROS node, with message-passing between the modules, as shown on Figure 1. The full control system works in real-time, with one steering command per frame, at 30 fps frame rate.

The Navigability Map Generator was created after collecting and classifying data on 15 frames of the environment. So, the ANN ensemble was generated, trained and included into the embedded system.

B. State Detection Unit

The ANN used for state detection was implemented and trained with Stuttgart Neural Network Simulator (SNNS) software. ANN training database was generated collecting about 2040 frames (converted to navigability maps) for each class. So, the final database was composed by 8165 input/output pairs.

The training algorithm used was Resilient Propagation (R-Prop). This algorithm achieves great results for feed-forward networks in many applications due to its good training time and convergence. Training parameters used are shown on Table 1.

Table 1 – Training Parameters

Parameter	Value
Training Algorithm	R-Prop
δ0	0,1
δMax	50
α	4
cicles	500
aprox. total training time	2 hours

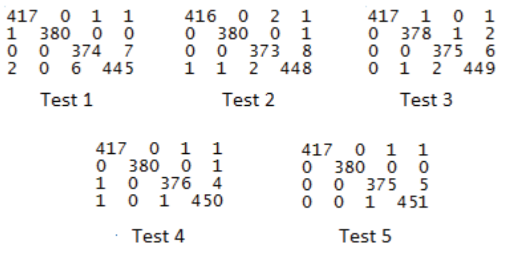
Due to the camera position, the first 288 elements on upper half of navigability matrices are most commonly related to elements above the horizon line, so they are not considered for state detection. The lower 96 elements (bottom 3 lines) are also discarded because they are related to the current navigable area, not an incoming situation. This way, a detection window with 12 lines (384 elements) is used for state detection. So, the ANN input layer is composed by 384 neurons only.

Some empirical tests based on previous work knowledge were performed in order to determine the best ANN topology. The best results were achieved by a feed-forward MLP, with the 384 input neurons, 384 neurons on hidden layer and 4 neurons on output layer (1 neuron per class), all neurons with activation function defined as "Act_Logistic" implemented on SNNS.

ANN validation was done with stratified 5-fold cross-validation method. This way, 5 train and test sets were generated, with 80-20 proportion on data (80% used for training and 20% for test, with same proportion of elements from the 4 classes on the datasets). The ANN accuracy on the five tests can be observed at Table 2, and the confusion matrices for the five test sets resulting from 5-fold cross validation are shown on Figure 8.

Table 2 – ANN Accuracy after 500 training cicles

	Test 1	Test 2	Test 3	Test 4	Test 5
ANN Accuracy	98.8%	99.0%	99.1%	99.3%	99.5%



417	0	1	1
1	380	0	0
0	0	374	7
2	0	6	445

Test 1

416	0	2	1
0	380	0	1
0	0	373	8
1	1	2	448

Test 2

417	1	0	1
0	378	1	2
0	0	375	6
0	1	2	449

Test 3

417	0	1	1
0	380	0	1
1	0	376	4
1	0	1	450

Test 4

417	0	1	1
0	380	0	0
0	0	375	5
0	0	1	451

Test 5

Figure 8 – Confusion matrices

A low error per class can be observed on confusion matrices. However, this only shows the network learned the database examples. If the system wrongly detects any situation as the next expected state, the reactive behavior will be affected, adding a bias to wrong templates and performing wrong steering actions. Likewise, if a new expected state is not detected in time, the system will keep the current behavior active, impairing self-localization and navigation tasks.

Therefore, state detection is a critical point. To ensure a correct navigation, the ANN training database must cover an enough amount of examples, avoiding misclassification on state detections. This problem can also be reduced if a minimum amount of consecutive new state detections is required for state transitions. The experiments carried out with this filter showed an adequate reliability level for state changes in developed applications.

C. *Autonomous Navigation Tests*

The environment in which the database was generated and the tests performed is shown next, on Figure 9. Our goal was to autonomously navigate following well-defined routes. The vehicle successfully accomplished the navigation task in a path with straights and a roundabout in autonomous mode, validating the proposed approach.

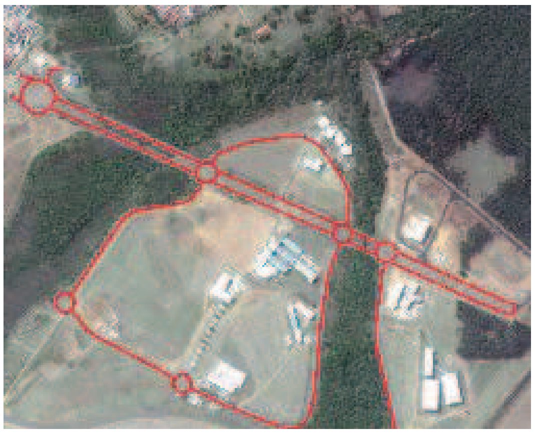


Figure 9 – Outdoor Environment used for tests

The reactive control was correctly affected by state detection, allowing a smooth behavior. Figure 10 shows the vehicle performing a left turn in a roundabout. A video with a successful path sequence and more information is available at http://www.icmc.usp.br/~fosorio/research/vicomor12.html.

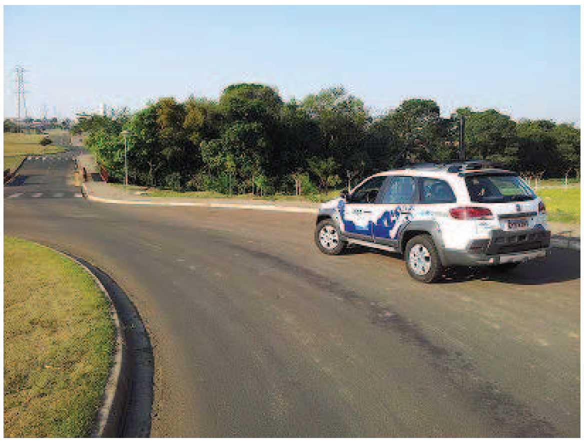


Figure 10 – Autonomous vehicle in a roundabout situation

Some misclassification occurred when unexpected light conditions affected input data, resulting in noisy navigability maps. Other dynamic elements such as vehicles and obstacles were not considered as a new state, and also produced bad responses when crossing the state detection window area.

The minimum amount of consecutive new state detections was manually tuned, normalizing the chance of straight and turn behaviors to be activated, since turn states are detected for a shorter period of time.

We also considered using a binary navigability map instead of continuous probability maps in template-matching step, filtering uncertainty on road detection with a threshold. In some cases, this solution produced better results than continuous version, so the use of binary maps was implemented as a parameter to be set before navigation system launch, allowing the selection of the fittest solution.

V. Conclusion

The successful navigation results demonstrated the suitability of this approach for autonomous vehicles navigation, when an accurate state detection is possible. This way, the use of a camera as the main sensor can be an efficient and reliable solution, allowing the project of low-cost autonomous driving systems.

This work complements the range of applications proposed on our previous indoor works, showing Topological Navigation as a promising approach for autonomous vehicles navigation too.

The system can be re-trained to recognize new situations, settings and features, and also use and combine other sensorial systems, allowing its implementation in different scenarios. For future works, we consider using sensor fusion and other techniques to detect new features and landmarks useful for navigation control.

ACKNOWLEDGMENT

The author acknowledges FAPESP and CNPq for their support to INCT-SEC (National Institute of Science and Technology - Critical Embedded Systems - Brazil), processes 573963/2008-9 and 08/57870-9 and financial support to authors (master´s 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. In: 7th IFAC Symposium on Intelligent Autonomous Vehicles, Lecce, 2010.

[4] Sales, D; Osório, F; Wolf, D. Topological Autonomous Navigation for Mobile Robots in Indoor Environments using ANN and FSM. In: Proceedings of the I Brazilian Conference on Critical Embedded Systems (CBSEC), São Carlos, Brazil, 2011.

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

[6] Marino, A.; Parker, L.; Antonelli, G. and Caccavale, F. Behavioral Control for Multi-Robot Perimeter Patrol: A Finite State Automata approach. In: ICRA, 2009.

[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.

[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] Heinen, Milton Roberto ; Osório, Fernando S. ; Heinen, Farlei ; Kelber, Christian . SEVA3D: Using Artificial Neural Networks to Autonomous Vehicle Parking Control.. In: IJCNN - IEEE Intenational Joint Conference on Neural Networks, 2006, Vancouver. Proceeding of the WCCI (World Congress on Computational Intelligence) - IJCNN. Vancouver, Canadá : IEEE Press, 2006. v. 1. p. 9454-9461.

[13] Pomerleau, D. ALVINN: An Autonomous Land Vehicle In a Neural Network. Advances in Neural Information Processing Systems 1, 1989.

[14] Pomerleau, D. RALPH: Rapidly Adapting Lateral Position Handler. IEEE Symposium on Intelligent Vehicles, September, 1995, pp. 506 - 511.

[15] Souza, J.; Sales, D. O.; Shinzato, P. Y.; Osório, F. S.; Wolf, D. F. Template-based autonomous navigation in urban environments. In: Proceedings of the 2011 ACM Symposium on Applied Computing, TaiChung, China, 2011.

[16] Sales, D. O.; Correa, D. S. O.; Osório, F. S.; Wolf, D. F. 3D Vision-based Autonomous Navigation System using ANN and Kinect Sensor. In: Conference Proceedings EANN 2012 – CCIS: Volume number 311., London, UK, 2012.

[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] Haykin, S. Neural Networks: A Comprehensive Foundation. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1998.

[20] Sales, D; Shinzato, P; Pessin, G; Wolf, D; Osório, F. Vision-based Autonomous Navigation System using ANN and FSM Control. In: Proceedings of the IEEE Latin American Robotics Symposium (LARS), São Bernardo do Campo, Brazil, 2010.

[21] Giles, C. Lee; Horne, Bill G.; LIN, Tsungnan. Learning a class of large finite state machines with a recurrent neural network. Neural Networks 8(9): 1359-1365. 1995.

[22] Omlin, Christian W.; Giles, C. Lee. Constructing Deterministic Finite-State Automata in Recurrent Neural Networks. Jornal of the ACM 43(6): 937-972. 1996.

[23] Frasconi, Paolo; Gori, Marco; Maggini, Marco; Soda, Giovanni. Representation of finite state automata in Recurrent Radial Basis Function networks. Machine Learning 23:1, 5-32 1996.

[24] Cleeremans, Axel; Servan-Schreiber, David; McClelland, James L. Finite State Automata and Simple Recurrent Networks. Neural Computation, Vol. 1, No. 3, Pages 372- 381. 1989.

[25] Sales, D; Feitosa, D; Osório, F; Wolf, D. Multi-Agent Autonomous Patrolling System using ANN and FSM Control. In: Proceedings of the II Brazilian Conference on Critical Embedded Systems (CBSEC), Campinas, Brazil, 2012.

[26] PointGray Bumblebee2 Stereo Cam. Internet: http://www.ptgrey.com/products/stereo.asp [09/2012]

[27] Robot Operating System (ROS). Internet: http://www.ros.org/wiki/ [09/2012]