Template-based autonomous navigation and obstacle avoidance in urban environments

Jefferson R. Souza, Daniel O. Sales, Patrick Y. Shinzato,
Fernando S. Osório and Denis F. Wolf
Mobile Robotics Laboratory, University of São Paulo (USP)
Av. Trabalhador São-Carlense, 400 - P.O. Box 668 - 13.560-970, São Carlos, Brazil
{jrsouza, dsales, shinzato, fosorio, denis}@icmc.usp.br
http://www.lrm.icmc.usp.br/

ABSTRACT

Autonomous navigation is a fundamental task in mobile robotics. In the last years, several approaches have been addressing the autonomous navigation in outdoor environments. Lately it has also been extended to robotic vehicles in urban environments. This paper presents a vehicle control system capable of learning behaviors based on examples from human driver and analyzing different levels of memory of the templates, which are an important capability to autonomous vehicle drive. Our approach is based on image processing, template matching classification, finite state machine, and template memory. The proposed system allows training an image segmentation algorithm and a neural networks to work with levels of memory of the templates in order to identify navigable and non-navigable regions. As an output, it generates the steering control and speed for the Intelligent Robotic Car for Autonomous Navigation (CaRINA). Several experimental tests have been carried out under different environmental conditions to evaluate the proposed techniques.

Categories and Subject Descriptors

I.2.9 [**Artificial Intelligence:Robotics**]: Autonomous Vehicles.

General Terms

Algorithms, Performance, Design, Experimentation.

Keywords

Robotic Vehicles Navigation, Obstacles Avoidance, Template Matching, FSM and Urban Environments.

1. INTRODUCTION

Human driver errors are a major cause of accidents on roads. Frequently people get injured or even die due to road traffic accidents (RTA). Also, bad road and weather conditions increase the risk of RTA. Autonomous vehicles could provide safer conditions in roads for individual or collective use. They also could increase efficiency in freight transportation and provide some degree of independence to people unable to drive.

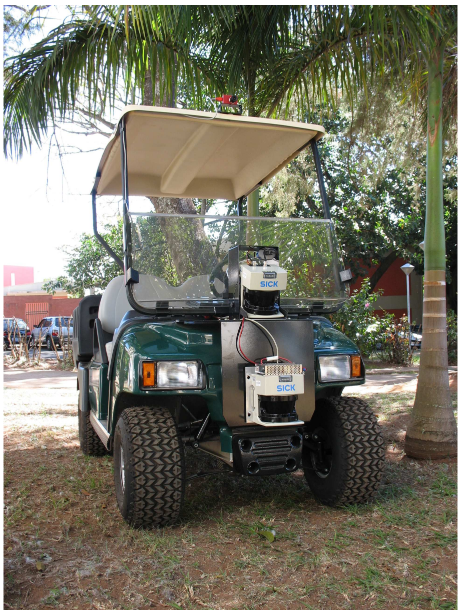
Figure 1: CaRINA test platform.

Research in mobile robotics has reached significant progress in the last 10 years. Part of them focus on autonomous navigation, which is a fundamental task in the area [19]. Lately, several works have been improving on navigation in outdoor environments. Competitions like DARPA Challenges [6] and ELROB [3] have been pushing the state of the art in autonomous vehicle control.

The most relevant results obtained in such competitions combine information obtained from a large number of complex sensors. Some approaches use five (or more) laser range finders, video cameras, radar, differential GPS, and inertial measurement units [6], [12]. Although there are several interesting applications for such technology, the cost of such systems is very high, which is certainly prohibitive to commercial applications.

In this paper we propose a vision-based navigation approach for urban environments, based on a low cost platform.

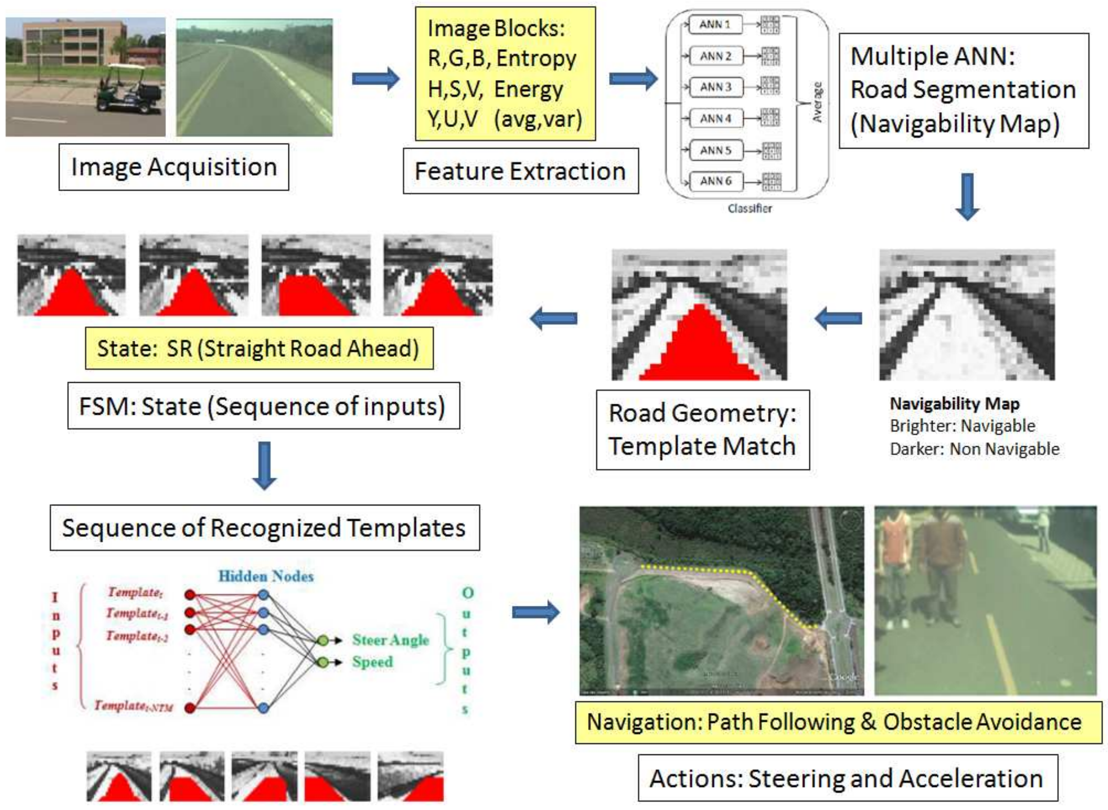
Image Acquisition
Image Blocks: R,G,B, Entropy H,S,V, Energy Y,U,V (avg,var)
Feature Extraction
ANN 1
ANN 2
ANN 3
ANN 4
ANN 5
ANN 6
Average
Classifier
Multiple ANN: Road Segmentation (Navigability Map)
Navigability Map
Brighter: Navigable
Darker: Non Navigable
Road Geometry: Template Match
State: SR (Straight Road Ahead)
FSM: State (Sequence of inputs)
Sequence of Recognized Templates
Inputs
Hidden Nodes
Outputs
Steer Angle
Speed
Navigation: Path Following & Obstacle Avoidance
Actions: Steering and Acceleration

Figure 2: General outline of the autonomous navigation system.

Our system uses a single camera to acquire data from the environment. It detects the navigable regions (roads), estimates the more appropriate maneuver, acquires and trains different levels of memory of the templates that should be done in order to keep the vehicle in a safe path, and finally, control steering and acceleration of the vehicle. Figure 1 presents our CaRINA test platform.

Our approach is based on two Artificial Neural Networks (ANNs). The first one identifies navigable regions in which a template-based algorithm classifies the image and identifies the action that should be taken by CaRINA. The images are acquired and then processed using ANNs that identifies the road ahead of the vehicle. After that, a Finite State Machine (FSM) is used to filter some input noise and reduce classification and/or control errors. In this work noise is considered as variations in the road color, such as dirt road (mud or dust), shadows, and depressions. After obtaining the current state (template), which is the input of a new ANN that works with levels of memory of the templates (LMT). This ANN aims to learn the driver's behavior, providing smoother steering and levels of speed in the same way as the driver. We considered six levels of template memory on the ANN searching to obtain the topology which provides more reliable results.

A sequence of states associated with an action is learned by the second ANN. States comes from situations identified by the Templates and FSM, and are mapped into steering angle and vehicle speed. Figure 2 shows a general outline of this system.

2. RELATED WORKS

Autonomous Land Vehicle in a Neural Network (ALVINN) [13] is an ANN based navigation system that calculates a steer angle to keep an autonomous vehicle in the road limits. In this work, the gray-scale levels of a 30 x 32 image were used as the input of an ANN. In order to improve training, the original road image and steering were generated, allowing ALVINN to learn how to navigate in new roads. The disadvantages of this work are the low resolution of a 30 x 32 image (gray-scale levels) and the high computational time. The architecture has 960 input units fully connected to the hidden layer to 4 units, also fully connected to 30 units in output layer. Regarding that issue, this problem requires real time decisions therefore this topology is not efficient.

Later, the EUREKA project Prometheus [7] for road-following was successfully performed using image based solutions, which provided trucks with an automatic driving system to reproduce drivers in repetitious long driving situations. The system also included a function to warn the driver in dangerous situations. A limitation of this project was an excessive number of heuristics created by the authors to limit the false alarms caused by shadows or discontinuities in the color of the road surface.

In the work [8], an outdoor mobile robot learns avoiding collisions by observing a human driver, the test platform is a vehicle equipped with sensors (laser, GPS and IMU) that produce a map of the local environment. This approach presents a method for automatically learning the parameters of

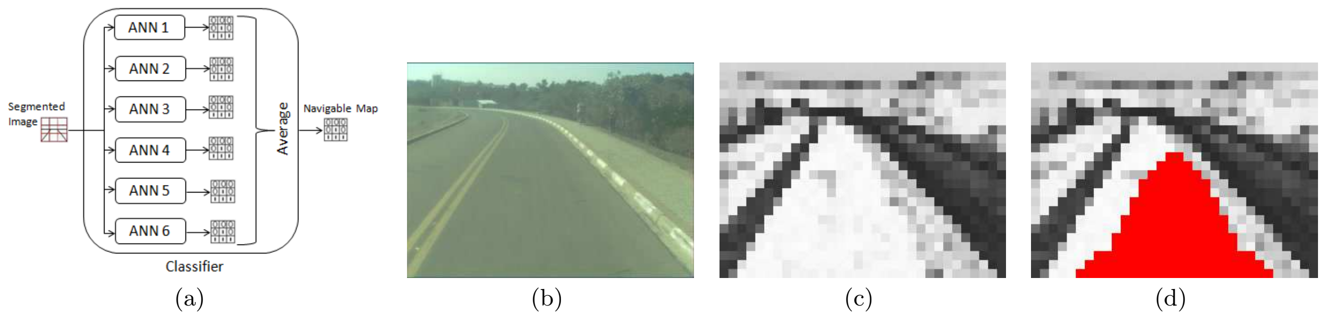

Figure 3: Classifier structure (a), real image (b), image processing step (c) and template matching (d).

the control model. The input to the control model is a goal point and a set of obstacles defined as points in the x-y plane of the vehicle. Three different approaches (Random, Genetic Algorithm (GA) and Simulated Annealing (SA)) were used in the learning step. The GA and Random performed similarly well. SA was worse than the other techniques. Also the authors compared the randomly learned parameter set with the principal component analysis, and observed than the random generalized well. The issue of this approach is the dependency of goal point and the set of obstacles, and all the parameters were limited to a fixed range.

Chan et al. [5] shows an Intelligent Speed Adaptation and Steering Control that allows the vehicle to anticipate and negotiate curves safely. This system uses Generic Self-Organizing Fuzzy Neural Network (GenSoFNN-Yager) which include the Yager inference scheme [11]. GenSoFNN-Yager has as main feature their ability to induce from low-level perceptual information in form of fuzzy *if-then* rules. Results show the robustness of the system in learning from example human driving negotiating new unseen roads. The autonomous driver demonstrate that anticipation is not always sufficient. Moreover, large variations in the distribution of the rule were observed, which imply a high complexity of the system.

Shihavuddin et al. [14] approaches the path map generation of an unknown environment using a proposed trapezoidal approximation of road boundary. At first, a blind map of the unknown environment is generated in computer, and then the image of the unknown environment is captured by the vehicle and sent to the computer using a RF transmitter module. After that, the image is pre-processed and the road boundaries are detected using the trapezoidal approximation algorithm. During this process, the vehicle operates independently avoiding the obstacles. The issue with this approach is the dependency of the camera tilt angle, because the vehicle moves through of the trapezium and reaches the next approximated trapezium having a previously tilt angle.

The work [10] focus on the task of lane following, where a robot-car learns anticipatory driving from a human and visual sensory data. During the learning steps the robot associates visual information with human actions. This information is derived from the street lane boundary that is detected in each image in real-time. In this work two modules were used, a reactive controller (RC) and a planner, where the former generates maps short-term information to a single steering control value, and the latter generates action plans, i.e. sequences for steering and speed control. The final steering command is a combination of planner and RC output. The advantages of this approach are react to upcoming events, cope with short lacks of sensory information, and use these plans for making predictions about its own state, which is useful for higher-level planning. Despite many advantages, due to the inertia of the robot it is less visible than what could be expected from the plotted signal. Also the system is not able to predict future states.

Stein and Santos [18], proposes a method to compute the steering of an autonomous robot, moving in a road-like environment. The proposed system used ANNs to learn behaviors based on examples from human driver, replicating and sometimes even improving human-like behaviors. To validate the created ANNs, real tests were performed and the robot successfully completed several laps of the test circuit showing good capacities for recovery and for generalization with relatively small training data sets. One of the issues in this work is the impossibility of validating network training without actually testing it with the real robot.

Markelic et al. [9], proposes a system that learns driving skills based on a human teacher. Driving School (DRIVSCO) is implemented as a multi-threaded, parallel CPU/GPU architecture in a real car and trained with real driving data to generate steering and acceleration control for road following. Besides, it uses an algorithm for detecting independently moving objects (IMOs) for spotting obstacles with stereo camera. A predicted action sequence is compared to the driver actions and a warning is issued if they are differing too much (assistance system). The IMO detection algorithm is more general in the sense that it will respond not only to cars, but to any sufficiently large (11 x 11 pixels) moving object. The steering prediction is very close to the human signal, but the acceleration is less reliable.

3. PROPOSED METHOD

Our approach is composed by 4 steps. In the first step an image is obtained and the road is identified using ANNs (Figure 3 (c)). In the second step, a template matching algorithm is used to identify the geometry of the road ahead of the vehicle (straight line, soft turn, or hard turn). In the third step, a FSM is used to filter noisy inputs and any classification error. Finally, a template memory is used in order to define the action that the vehicle should take to keep on road. These steps will be described in the next sub-sections.

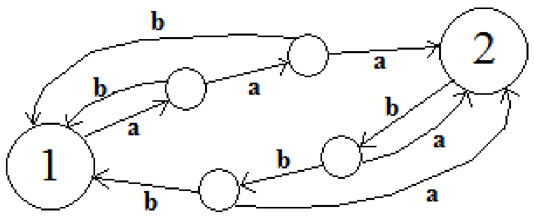

Figure 4: Transition between 2 states with 2 intermediate states.

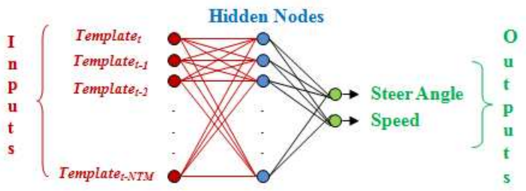

Figure 5: Structure of the second ANN used to generate steering and acceleration commands.

3.1 Image Processing Step

We adopted the proposed method of Shinzato [15], which proposes to use ANNs to be applied into a road identification task. Based on the results, a system composed by six Multilayer Perceptron (MLP) ANNs was proposed to identify the navigable regions in urban environments (Figure 3 (a)). The real image can be seen on Figure 3 (b). The result of this ANNs output combination is a navigability map (Figure 3 (c)). The brighter blocks are the more likely area to be considered navigable. This step divides an image into blocks of pixels and evaluates them as single units. The advantage of this approach is that one can train the ANNs to identify different types of navigable and non-navigable regions (e.g. pavemented, non-pavemented roads, sidewalks).

Initially, the image processing step divides the image into blocks of pixels and evaluates then as single units. Several features are calculated for each block, such as: pixel attributes like red, green, blue (RGB) average, image entropy and others features obtained from this collection of pixels (region block). In the grouping step, a frame with (MxN) pixels resolution was sliced in groups with (KxK) pixels. Suppose an image represented by a matrix I of size (MxN). The element *I(m,n)* corresponds to the pixel in row m and column n of image, where ($0 <= m < M$) and ($0 <= n < N$). Therefore, group *G(i,j)* contains all the pixels *I(m,n)* such that $((i*K) <= m < ((i*K)+K))$ and $((j*K) <= n < ((j*K)+K))$. This strategy has been used to reduce the amount of data, allowing faster processing.

Once a block is processed, its attributes are used as inputs of the ANNs. The ANNs are used to classify the blocks considering their attributes (output 0 to non-navigable and 1 to navigable). Each ANN contains an input layer with the neurons according to the image input features (see Table 1), one hidden layer with five neurons, and the output layer has only one neuron (binary classification). However, after the training step, the ANN returns real values between 0 and 1, as outputs. This real value can be interpreted as the classification certainty degree of one specific block. The main difference between the six ANNs is the set of image attributes used as input. All these sets of attributes (see Table 1) are calculated during the block-segmentation of the image. The choice of these attributes was based on the results presented in the work [15].

Table 1: Input attributes of the ANNs (R, G, B = red, green, blue components; H, S, V = hue, saturation, value components; Y, U, V = Luminance, average = av, normalized = norm, entropy = ent, energy = en and variance = var).

ANNs	Input attributes
ANN1	U av, V av, B norm av, H ent, G norm en and H av
ANN2	V av, H ent, G norm en, G av, U av, R av, H av, B norm av, G norm av and Y ent
ANN3	U av, B norm av, V av, B var, S av, H av, G norm av and G norm ent
ANN4	U av, V av, B norm av, H ent, G norm en and H av
ANN5	V av, H ent, G norm en, G av, U av, R av, H av, B norm av, G norm av and Y ent
ANN6	U av, B norm av, V av, B var, S av, H av, G norm av and G norm ent

After obtaining the six outputs of the ANNs referring to each block, the classifier calculates the average of these values to compose a single final output value. These values representing each block obtained from the original image form together the navigability map matrix (Figure 3(c)). This matrix is used to locate the most likely navigable region. It is important to mention that the ANN is previously trained using supervised examples of navigable and non-navigable regions selected by the user one time on an initial image frame. After that, the trained ANN is integrated into the vehicle control system and used as the main source of information to the autonomous navigation control system.

3.2 Template Matching Step

After obtaining the ANN classification, 7 different road templates are placed over the image in order to identify the road geometry (some examples of these templates are presented in Figures 6, 7 and 8). One of them identifies a straight road ahead, two identify a straight road in the sideways, two identify soft turns, and two identify hard turns (e.g. a straight road ahead Figure 3 (d)). Each template is composed by a mask of 1s and 0s as proposed in [17]. The value of each mask is multiplied by the correspondent value into the navigability matrix (values obtained from the ANN classification of the correspondent blocks of the image). The total score for each template is the sum of products. The template that obtains the higher score is selected as the best match of the road geometry. Only one template can obtain a high score, because we use probabilities as the decision criteria.

3.3 Finite State Machine Step

The FSM uses the result of the template matching step as input, which carries out a classification for the road detected in each captured frame. This classification is defined by the template which best fits the matrix and its position. The developed FSM is composed by 5 states (straight road, soft turns left and right, and hard turns left and right) as shown on Table 2.

Table 2: States of the FSM used for the experimental tests.

States	Associated Action
SR	Keep steering wheel at center position
SLT	Smoothly turn the steering wheel to left
SRT	Smoothly turn the steering wheel to right
LT	Fully turn the steering wheel to left
RT	Fully turn the steering wheel to right

Figure 4 presents a partial scheme of the developed FSM. Transition 'a' represents classification of an input as a transition in direction of State 2, and 'b' classification as a transition in direction of State 1. Figure 4 represents a state change of state 1 to state 2. For example, 'a' represents a straight road state and 'b' soft turns left. To change the state in the FSM there must happen three consecutive equal states. In this work, we use the FSM with only 2 intermediate transitions between the states and have produced reasonable results. Detailed information can be seen in [17].

The full FSM was designed with 5 states, which is obviously more complex to represent, but the partial scheme presented in Figure 4 still remains representative of the full FSM scheme. A transition between the current states to any other state occurs only after detecting a sequence of 'n + 1' identical inputs leading to the new state, where 'n' is the established number of intermediate states. The number of intermediate states varies according to the noise level in the images and navigability matrix representing the road, and also depends of the frame rate. In a system based on a high image acquisition frame rate, more misclassified inputs could be generated per time unit, so more intermediate states can be needed to discard some of these bad/noisy inputs. Environments with low level of input noise are associated to a smaller number of intermediate states.

3.4 Template Memory Step

After obtaining the current state (template) by FSM, this current template is used as input in the template memory step. In this step, the levels of memory of the templates are stored in a queue, as $\{Template_t,\ Template_{t-1},\ Template_{t-2},\ ...,\ Template_{t-NTM}\}$. In this work, the $Template_t$ represents the current template, $Template_{t-1}$ the previous template, $Template_{t-2}$ one template before the previous. This is done successively, until the number of template memory (NTM) is reached, where t represents the time.

In this step, a second ANN is used differently of the ANNs used in the image processing step. The basic network structure (Figure 5) used is a feed-forward MLP, the activation function of the hidden neurons is the sigmoid function and the ANN learning is the resilient backpropagation (RPROP). The inputs are represented by templates memory and the outputs are the steer angle and speed.

4. EXPERIMENTAL RESULTS

The experiments were performed using CaRINA (Figure 1), an electric vehicle capable of autonomous navigation in a urban road, equipped with a VIDERE DSG video camera, a ROBOTEQ AX2580 motor controller for steering control, an ARDUINO DUEMILANOVE is a microcontroller board used for vehicle speed control and a GARMIN 18X-5Hz GPS. The GPS was used only to register the log of the vehicle trajectories, and it was not used as input of the system. The image acquisition resolution was set to (320 x 240) pixels. The ANNs of the image processing step were executed using Fast Artificial Neural Network (FANN) [1], which is a free open source library which implements MLP ANNs in C, and the ANN in the template memory step was used Stuttgart Neural Network Simulator (SNNS) [2], which is also a software simulator for ANNs. For the development of the image acquisition, image processing and template matching algorithm, we used the OpenCV [4] library.

The results are divided into three scenarios. The first one shows an urban area without obstacles using only the steering control without template memory. The second scenario shows an urban area with obstacles (e.g. people, traffic cones) using the steering and speed control without template memory. The third scenario shows an urban area without obstacles, but uses the supervised learning (vehicle control system capable of learn behaviors based on examples obtained from human driver).

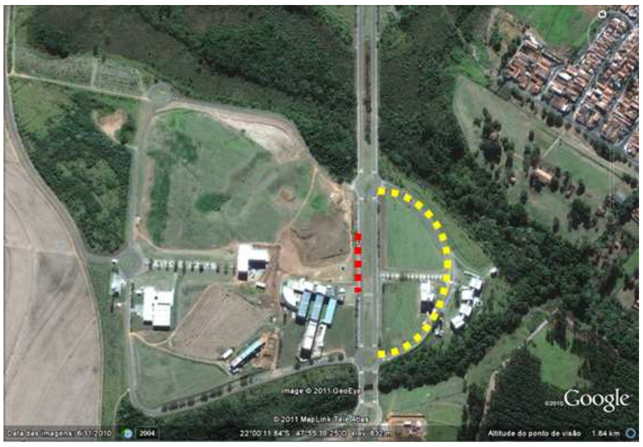

Figure 10: GPS trajectories (Scenario 1).

The results of the first scenario are described in two experiments performed with our vehicle[1]. In the first one, the vehicle should follow the road pavement (asphalt surface), including straight, soft turn and hard turn path segments. In the second experiment the vehicle should navigate into a narrow straight path segment, following a surface composed by a red pavement (red bricks). Both trajectories can be seen in Figure 10 (yellow lane - 1st Experiment and red lane - 2nd Experiment). The first experiment is the longer path, and the second one is the shorter and narrower straight path, both described in terms of their absolute GPS coordinates. In both cases the vehicle was able to keep itself in the road

[1]Experiment videos available in the Internet (Scenario 1):
1st. - http://www.youtube.com/watch?v=boJ_jRmtclU/
2nd. - http://www.youtube.com/watch?v=aJd31XoL6vA/

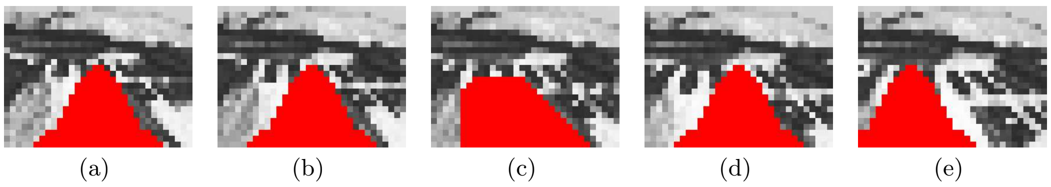
Figure 6: Road identification. (a), (b), (d), and (e) are straight road. (c) is soft left turn.

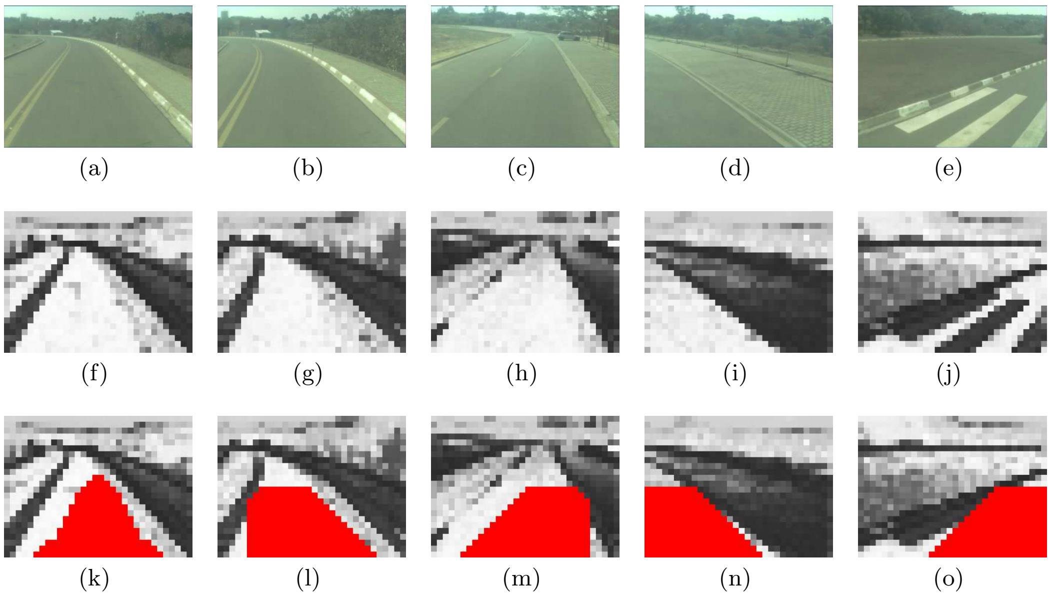
Figure 7: Experimental results of Scenario 1 (first experiment). Original Image (a), (b), (c), (d) and (e). Classification by ANNs (f), (g), (h), (i) and (j). Command output (k), (l), (m), (n) and (o).

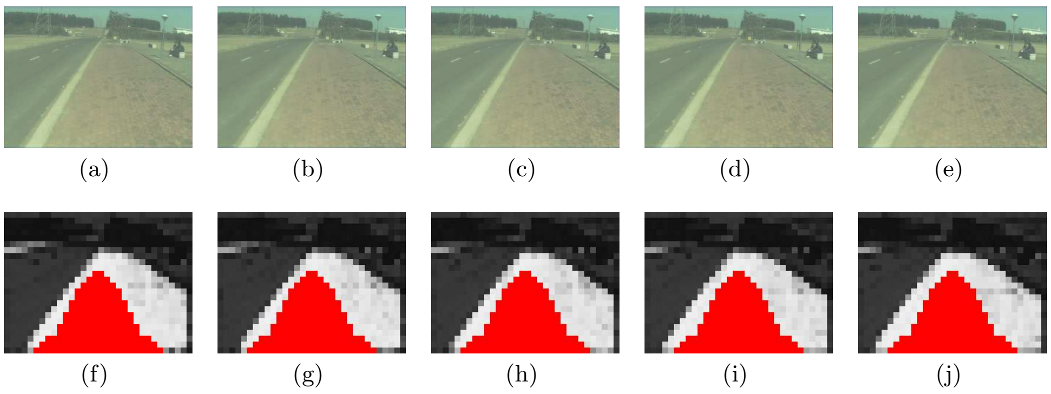
Figure 8: Experimental results of Scenario 1 (second experiment). Original Image (a), (b), (c), (d) and (e). Classification by ANNs (f), (g), (h), (i) and (j).

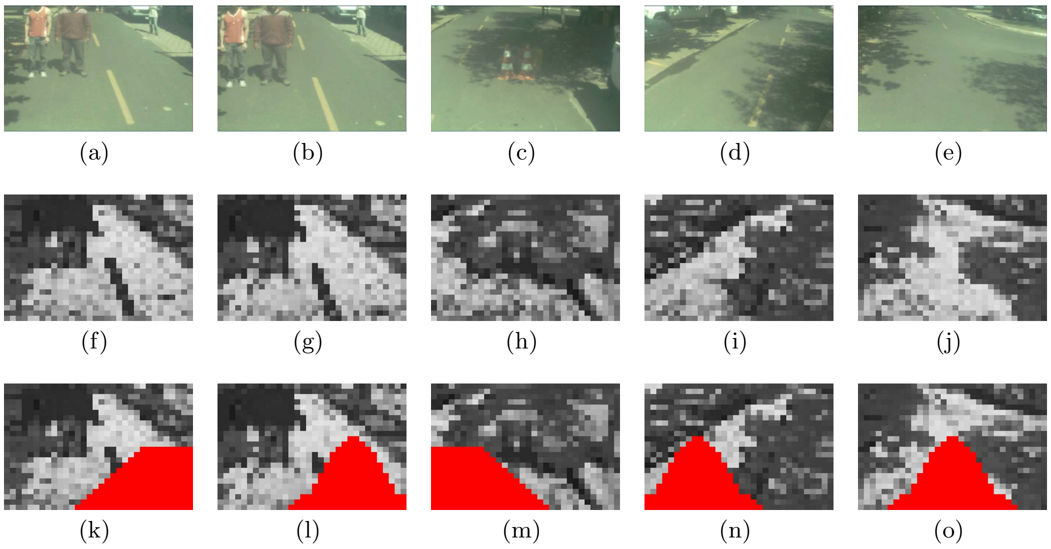

Figure 9: Experimental results of Scenario 2. Original Image (a), (b), (c), (d) and (e). Classification by ANNs (f), (g), (h), (i) and (j). Command output (k), (l), (m), (n) and (o).

successfully, following the road path defined by the region previously indicated and trained as the navigable surface.

In order to obtain a quantitative analysis of the system performance, 30 representative frames of the defined states for this problem have been manually classified and compared to the outputs of the proposed algorithms. As shown in Table 3, satisfactory results were obtained (67.4% overall correct classification result). Most errors are due to small variations in the classification, and occurred between neighbor states/templates: left turn (LT) and soft left turn (SLT); right turn (RT) and soft right turn (SRT).

Table 3: Steering command results from Scenario 1 (first experiment).

States	SR	LT	RT	SLT	SRT
SR	**93.4%**	0.0%	0.0%	6.6%	0.0%
LT	20.0%	**53.4%**	0.0%	26.6%	0.0%
RT	13.3%	0.0%	**83.4%**	0.0%	3.3%
SLT	46.6%	0.0%	0.0%	**53.4%**	0.0%
SRT	46.6%	0.0%	0.0%	0.0%	**53.4%**

Figure 6 shows an example of the proposed identification method that resulted in 15 total state transitions (ST) detections (considering only the present frame), during a total sequence of 60 frames. The proposed FSM reduced these 15 transitions to only 4 effective ST. One example of a wrong frame discarded by FSM is showed in Figure 6 (c), where the current state was kept. Even when traversing a crosswalk the vehicle successfully keep moving in the correct direction.

Figure 7 presents an example of straight, soft left and right turns, and hard left and right turns being accurately identified by our system, with successfully navigation inside the safe region defined by the pavement paths (navigable regions). Figure 8 shows an example of a straight path being accurately identified by our system, with successfully navigation following the path defined by the red bricks surface.

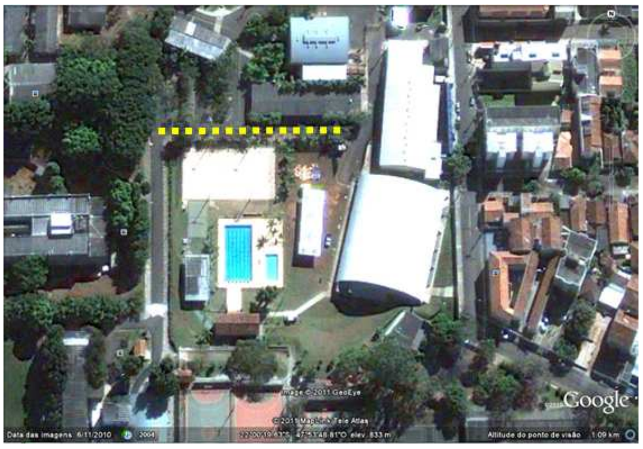

Figure 11: GPS trajectory (Scenario 2).

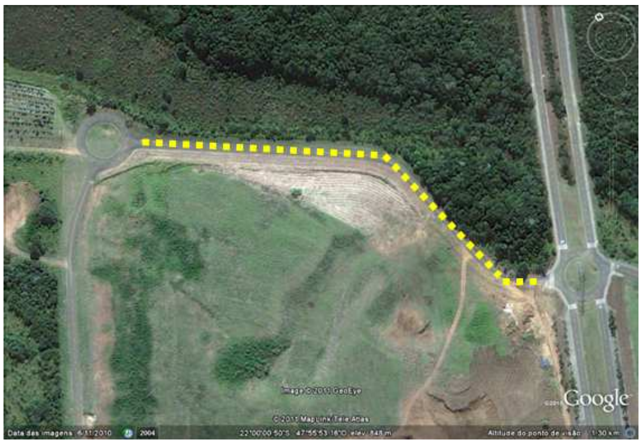

Figure 12: GPS trajectory (Scenario 3).

The results of the second scenario were performed with the CaRINA platform[2]. The vehicle should keep on the urban

[2]Experiment video available in the Internet (Scenario 2):

road, spotting of obstacles (e.g. people, traffic cones), including straight, soft turn and hard turn path segments. The GPS trajectories using the vehicle can be seen in Figure 11.

Figure 9 shows in details the experiment of scenario 2, including an example of straight, soft left and right turns, and hard left and right turns being identified by our system, performing successfully the navigation inside the safe region defined by the urban path and the obstacles avoidance (Figure 9 (a), (b) and (c)).

The results of the third scenario also were performed with the CaRINA platform [16]. The vehicle should keep on the urban environment (road), replicating the human behavior based on examples from human driver. The GPS trajectory can be seen in Figure 12.

Table 4 shows the obtained values of the performed path by CaRINA for supervised learning. In Table 4, we analyze six LMT, which represent the architecture of the second ANN used in our proposed system. The numbers of LMT column represents the values that we tested randomly in order to develop a well-defined architecture. Half, Equal and Double shows the different architectures, for example, LMT = 3, where changes occur in the number of neurons in the intermediate layer of a MLP architecture RPROP (a learning heuristic for supervised learning in feed-forward ANNs), tested the architectures: 3-1-2 (Half), 3-3-2 (Equal) and 3-6-2 (Double) (half the size of the ANN input layer equal the size of the ANN input layer double the size (in neurons) of the ANN input layer), obtaining the values of mean squared error (MSE), epoch (optimal point of generalization (OPG), i.e., minimum training error and capacity of maximum generalization) and the number of best ANN (because were initialized 5 different random seeds).

Table 4: Best results from MSE (M), Epoch (E) and ANN (A) for the six levels of template memory.

LMT	Half			Equal			Double		
	M	E	A	M	E	A	M	E	A
3	45.599	25	4	45.837	40	1	45.847	55	3
5	51.441	45	3	51.404	25	2	51.362	25	5
8	43.601	15	1	45.067	60	2	44.381	15	1
10	48.969	15	3	51.334	55	4	49.431	10	5
15	43.481	60	5	42.941	15	5	**42.233**	20	4
20	52.133	20	4	52.253	45	1	52.268	50	2

After the analysis of data on Table 4, the architecture 15-30-2 showed the lowest MSE for the epoch 20.

Table 5 presents the largest initial values of MSE for all LMT, based on Table 4 that shows the lowest values of MSE. Analyzing the best ANNs learned, reducing the MSE until the OPG. The architectures (3-1-2, 5-2-2, 8-4-2, 10-5-2, 15-7-2 and 20-40-2) showed the largest initial values of the MSE in order a reduction of high MSE.

Table 6 shows the percentage of the MSE for all architectures of the second ANN proposed, based on Tables 4 and 5. The best architecture (15-30-2) is shown on Table 6, presenting the approximately error 5.9%. The lowest MSE of the Table

http://www.youtube.com/watch?v=U2rRzNUPU8Y/

Table 5: Best initial values of MSE.

	MSE		
LMT	Half	Equal	Double
3	**1060.730**	929.475	762.547
5	**1004.350**	825.228	606.680
8	**858.441**	662.772	463.422
10	**847.994**	622.916	472.747
15	**714.645**	467.323	693.573
20	603.171	463.863	**1385.200**

6 is defined as being the architecture 20-10-2 (error 3.7%), but after analyzing the data on Table 4 of this architecture, showed a high MSE (52.133) compared to other architectures. Therefore, the architecture 15-30-2 is the best.

Table 6: MSE results for LMTs.

	MSE		
LMT	Half	Equal	Double
3	4.298%	4.321%	4.322%
5	5.121%	5.118%	5.114%
8	5.079%	5.250%	5.170%
10	5.774%	6.053%	5.829%
15	6.084%	6.008%	5.909%
20	**3.763%**	3.772%	3.773%

The Figures 13 and 14 illustrate the steer angle and speed of CaRINA using the architecture 15-30-2, showing the comparison between human driver and the ANN learned. Small oscillations are present in the data ANN learned, since the FSM used in the proposed system maintains the current state during 2 intermediate transitions, resulting in a linearity of the data (the vehicle keeps on the road with steer and constants speed).

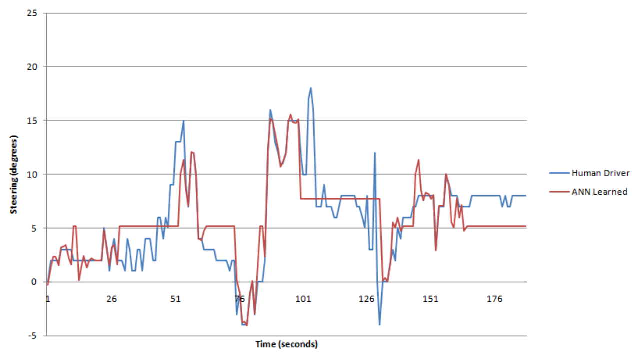

Figure 13: CaRINA steering - training data.

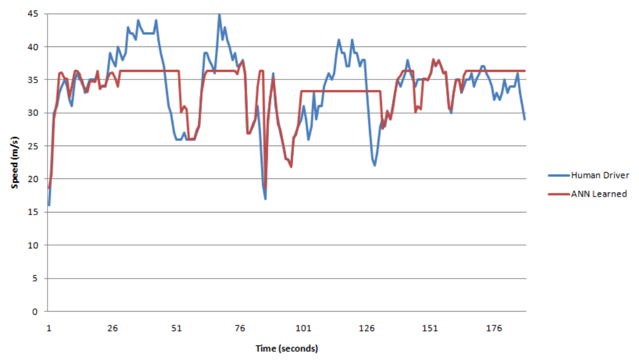

Figure 14: CaRINA speed - training data.

The Figures 15 and 16 illustrate two tests performed by CaRINA obtained of the second ANN used in our proposed system. The architecture 15-30-2 was used in our experiments, showing the steer angle and speed for the test data.

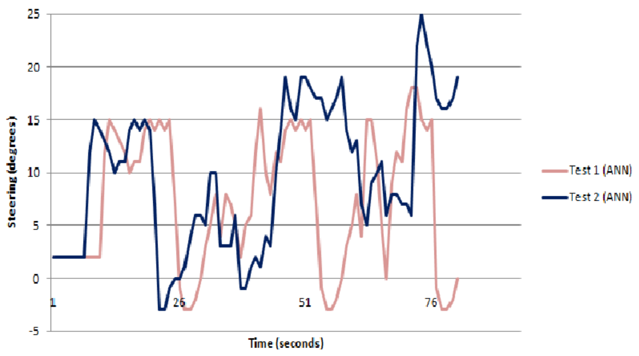

Figure 15: CaRINA steering - results obtained.

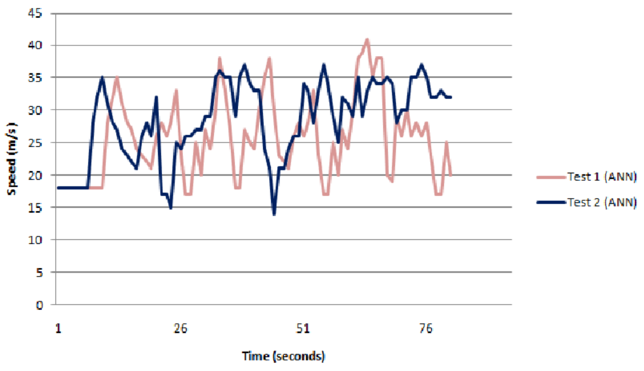

Figure 16: CaRINA speed - results obtained.

Figure 15 illustrates the steering performance of the ANN trained with the RPROP algorithm. It shows a better performance, with little oscillation as shown in Test 1 (the degree of steering between -5 and 15 as learned by ANN). The main difference when compared with Test 2 is small oscillation during the straight line used by CaRINA and it is quite similar the learned by the ANN. Unlike Test 2, the degree of steering is more than 15 in some parts.

The speed performance is shown on Figure 16 for the test data. In this case, the speed for Test 1 was obtained by ANN, showing a better performance when compared to Test 2, because it showed a better response to the learning of ANN and also generalized the data to navigate the vehicle on a safe path on the road unlike Test 2.

Experimental tests showed that the Test 1 generated as ANN output a better performance in the robot (CaRINA) behavior, as shown on Figures 15 and 16 (see Video[3]).

5. CONCLUSION AND FUTURE WORKS

Autonomous vehicle navigation is an important task in mobile robotics. This paper presented a vision-based navigation system which can be trained to identify the road and navigable regions using ANNs, template matching classification, FSM and a template memory. Our approach was evaluated using an Electrical Vehicle (CaRINA) tested in urban road following experiments. The vehicle was able to navigate autonomously in this environments executing a road following task, avoiding obstacles (e.g. people, traffic cones), and replicating the human behavior. Our quantitative analysis also obtained satisfactory results for the learning of ANNs with the respective architectures (LMT).

As future works, we plan to evaluate other classification methods and decision making algorithms. We also are planning to integrate camera and LIDAR laser in order to better deal with bumps and depressions in the road.

[3]Experiment video available in the Internet (Scenario 3): http://www.youtube.com/watch?v=SxdEY4JavAo/

6. ACKNOWLEDGMENTS

The authors acknowledge the support granted by CNPq and FAPESP to the INCT-SEC (National Institute of Science and Technology - Critical Embedded Systems - Brazil), FAPESP doctoral grant (process 2009/11614-4), and Gustavo Buzogany for the technical contributions.

7. REFERENCES

[1] Fast artificial neural network library (fann), 2010. http://leenissen.dk/fann/, Access on 02 June.

[2] Stuttgart neural network simulator (snns), 2010. www.ra.cs.uni-tuebingen.de/SNNS/, Access 20 Nov.

[3] European land-robot (elrob), 2011. http://www.elrob.org/, Access on 18 May.

[4] G. Bradski and A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. 2008.

[5] M. Chan, D. Partouche, and M. Pasquier. An intelligent driving system for automatically anticipating and negotiating road curves. *Int. Conf. on Intelligent Robots and Systems*, pages 117–122, 2007.

[6] H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun, and G. Bradski. Self-supervised monocular road detection in desert terrain. In G. Sukhatme, S. Schaal, W. Burgard, and D. Fox, editors, *In Proceedings of the Robotics Science and Systems Conference*, 2006.

[7] V. Graefe. Vision for intelligent road vehicles. *IEEE Symp. of Intell. Vehicles*, pages 135–140, 1993.

[8] B. Hamner, S. Scherer, and S. Singh. Learning to drive among obstacles. *IROS*, pages 2663–2669, 2006.

[9] I. Markelic, A. Kjaer-Nielsen, K. Pauwels, L. B. W. Jensen, N. Chumerin, A. Vidugiriene, M. Tamosiunaite, A. Rotter, M. V. Hulle, N. Kruger, and F. Worgotter. The driving school system: Learning automated basic driving skills from a teacher in a real car. *Trans. on Intell. Transp. Systems*, 2011.

[10] I. Markelic, T. Kulvicius, M. Tamosiunaite, and F. Worgotter. Anticipatory driving for a robot-car based on supervised learning. *In Lecture Notes in Computer Science: Anticipatory Behavior in Adaptive Learning Systems*, pages 267–282, 2009.

[11] R. J. Oentaryo and M. Pasquier. Gensofnn-yager: A novel hippocampus-like learning memory system realizing yager inference. *In International Joint Conference on Neural Networks*, pages 1684–1691.

[12] A. Petrovskaya and S. Thrun. Model based vehicle tracking in urban environments. *IEEE International Conference on Robotics and Automation*, 2009.

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

[14] A. S. M. Shihavuddin, K. Ahmed, M. S. Munir, and K. R. Ahmed. Road boundary detection by a remote vehicle using radon transform for path map generation of an unknown area. *Int. Journal of Computer Science and Network Security*, 8(8):64–69, 2008.

[15] P. Y. Shinzato and D. F. Wolf. A road following approach using artificial neural networks combinations. *Journal of Intelligent and Robotic Systems*, 62(3):527–546, 2010.

[16] J. R. Souza, G. Pessin, P. Y. Shinzato, F. S. Osório, and D. F. Wolf. Vision-based autonomous navigation using neural networks and templates in urban environments. *First Brazilian Conference on Critical Embedded Systems (I CBSEC)*, pages 55–60, 2011.

[17] J. R. Souza, D. O. Sales, P. Y. Shinzato, F. S. Osório, and D. F. Wolf. Template-based autonomous navigation in urban environments. *In 26th ACM Symp. on Applied Computing*, pages 1376–1381, 2011.

[18] P. S. Stein and V. Santos. Visual guidance of an autonomous robot using machine learning. *7th IFAC Symposium on Intelligent Autonomous Vehicles*, 2010.

[19] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L. E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. Van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney. Stanley, the robot that won the darpa grand challenge. *Journal of Field Robotics*, 2006.