2013 Fifth International Conference on Computational Intelligence, Communication Systems and Networks

Lanes Detection Based on Unsupervised and Adaptive Classifier

Andrés F. Cela
Department of Automation and Industrial Control
Escuela Politécnica Nacional
Quito, Ecuador
andres_cela@ieee.org

Luis M. Bergasa
Department of Electronics
University of Alcalá
Alcalá de Henares, España
bergasa@depeca.uah.es

Franklin L. Sánchez
Department of Electronics
Instituto Tecnológico Superior Sucre
Quito, Ecuador
fsanchez@tecnologicosucre.edu.ec

Marco A. Herrera
Center for Automation and Robotics
Universidad Politécnica de Madrid
Madrid, Spain
marco.herrera@ieee.org

***Abstract*— This paper describes an algorithm to detect the road lanes based on an unsupervised and adaptive classifier. We have selected this classifier because in the road we do not know the parameters of lanes, although we know that lanes are there, only they need to be classified. First of all, we tested and measured the brightness of the lanes of the road in many videos. Generally, the lines on the road are white. We used the HSV image and we improved the region of study. Then, we used a Hough transform which yields a set of possible lines. These lines have to be classified. The classifier starts with initial parameters because we suppose that the vehicle is on road and in the center of the lane. There are two classes, the first one is the left road line and the second one is the right road line. Each line has two parameters that are: middle point of line and the line slope. These parameters will be changing in order to adjust to the real lanes. A tensor holds the two lines, so these lines will not separate more than the tensor allows. A Kalman filter estimates the new class's parameters and improves the tracking of the lanes. Finally, we use a mask in order to highlight the lane and show to the user a better image.**

***Keywords*- Lanes detection, lanes classifiers, Kalman filter.**

I. INTRODUCTION

There is a high requirement in the car industries of implementing road lanes identifier systems on board. This is to give more safety to passengers. Because this, the vision-based lane detection has been an interesting research area. Nowadays, most of applications are make off-line.

There are many techniques for detecting the road lanes and most of them use vision with mono-camera. Although, using a stereo camera is being common. In this area there are two types of detections, the road detection and the lane road detection. The approaches in these fields in the last 5 years is surveyed in [1]. The DARPA grand challenge 2006 helped to make significant progress for commercial driver's assistance systems. In this challenge vehicles has a lot of sensors, to detect road, signs, edges, traffic lights, and of course the lane marks. However, researches have detected that the bottleneck in this systems is the road perception [1].

The researches in [2] show how to detect the road region based on homography estimation. They use a set of cameras and using a three module algorithm can detect the road. The work in [3] also uses a stereo camera for detecting the road with the 3D information. Using a stereo camera can give some advantages over a monocular camera, but in normal applications to use a monocular camera is cheaper. On the other hand, it is better to make low cost applications in order to implement them in smart phones, tablets or things like these, which will be our future work.

A more simple system is showed in [4]. This is based on a roads classifier which decides if the image is a road or not. This classifier has a data base of many kinds of roads.

Also, there are many models and researches to detect the road lanes. Only 8 % of the works are in real time algorithms. It is still complex to implement a real time algorithm, because for obtaining better results good and big processors are necessary.

Using the vanishing point is a way to find the lanes, for example [5]. This work has 94% of true detections, but it does not work in real time.

When a lane is not uniform or is a curve, it is more difficult to use the Hough transformation or the basic RANSAC. In [6] is presented a solution for curve lanes and in [7] a solution for heterogeneous lanes. The work in [8] is based on RANSAC to get the road lanes. This work uses a special camera which is able to collect 500 frames per second. This is used for high speed vehicles.

To detect the road lanes is necessary to consider the environmental conditions, like the light, rain, brightness, etc. For example, the researches in [9] present a solution for night conditions. This is based on the light reflections of the lines of the road. Then, a Support Vector Machine (SVM) classifies these lines.

In some cases it is necessary to know the Euler angles of the camera, like in [2], [5] y [8], but in this work we do not use these angles. First of all, these angles have to be

978-0-7695-5042-8/13 $26.00 © 2013 IEEE
DOI 10.1109/CICSYN.2013.40

IEEE computer society

calibrated and a common user does not have time to do that. Do not forget that we will implement this work in a smart phone in a future work. But, like other works, for example [9], our algorithm requires an initial condition. The camera should be focusing the road and the vanishing point has to be in the middle of the screen. We also used a Region of Interest (ROI) in order to reduce the information to process, [9] and [10].

To the user it is important to know the relative vehicle position in the road. However, this information must not distract the driver. We have implemented a simple visual effect that shows in the screen the vehicle deviation from the middle lane. It is not in meters but in percentage.

Our algorithm has three phases that are: The initial treatment of the image, the detection and lines classification and the filtering and image masked. Section 2 shows the proposed algorithm and section 3 shows the experimental result.

II. PROPOSED ALGORITHM

The algorithm architecture is done in three phases which are depicted in Fig. 1.

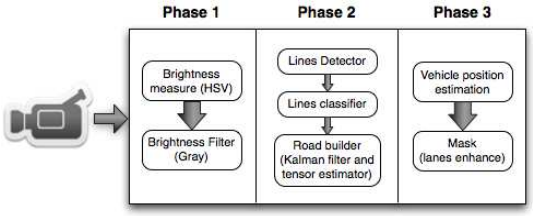

Figure 1. Algorithm architecture.

In Phase 1 we make a previous treatment of the input image. We measure the image brightness, then we define a ROI, and finally the new image is filtered to detect the lanes road in a binary image.

In Phase 2 a Hough transform is applied to the image. The result lines are classified by an unsupervised classifier. Then, a Kalman filter reduces the noise in the parameters of the lines of each class. These lines are showed in the original image and they built the road.

The vehicle position is estimated in the last phase and the real lanes of the road are highlighted. An indicator shows the driver the deviation on the screen.

A. Phase 1.Image preparation.

In this step the algorithm prepares the image to make the work simpler for the next phases. First, the image is transformed from RGB to HSV. Then, it calculates the brightness of the image which helps to know if it corresponds to day, afternoon or night. We have obtained some experimental results about the color of the lanes, which in most cases are black. The brightness lane is generally higher than the other colors in the image, it is 0.87/1. However, it is necessary to define a ROI in the lower half image. The brightness channel is the new image. Then, it is changed from HSV to Gray image with (1), where f is the RGB NxM input matrix image, (i,j) is the pixel position in i row and j column, V is the the V channel from the HSV image and G is the newer gray image.

In this step we proposed a new approach. Most of the researchers work with the RGB or Gray input, but we work with the brightness's image instead.

$$G(i,j) = HSV,V[f(i,j)] \quad (1)$$

$$Bm = (N*M)^{-1}\sum_{i=0}^{N/2}\sum_{j=0}^{M} G(i,j) \quad (2)$$

In (2) Bm is the brightness image average and it is calculates from the ROI. In the next phases we work in the same region.

We can find the lanes pixels with the filter presented in (4). This is based on the filter (3) proposed in [10], but it has been improved. In (4) we have added a contrast between the filter responses in each pixel and its brightness value. If the filter finds a lane pixel and the brightness is lower than a minimal, the pixel is a lane pixel. In (3) and (4), Im is the filtered image, I is the image after the improved filter and s is the number of pixels around the (i,j) pixel. The researches in [10] propose to try different values for s and to select the best results. In our case $s = 8p$.

$$Im(i,j) = 2*G(i,j) - G(i,j-s) - G(i,j+s) - |G(i,j-s) - G(i,j+s)| \quad (3)$$

$$I(i,j) = f(x) = \begin{cases} 0, & G(i,j) < Bm \\ Im(i,j), & G(i,j) \geq Bm \end{cases} \quad (4)$$

Figure 2 shows result from the steps in phase 1.

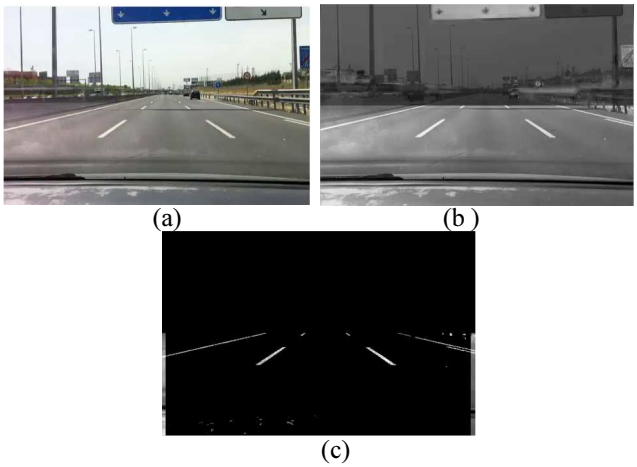

Figure 2. Phase 1 result. (a) Input image. (b) Gray brightness image. (c) Filtered image.

B. Phase2. Lines classifier and road builder.

The second stage in our algorithm consists of two steps that are: the lines classifier and the road builder.

The first step in this phase is to apply a Hough transform at the last filtered image which is showed in Fig. 2(c). We consider that this transformation is well known, so we do not

explain its functionality, being the most important that we obtain a set of lines which may represent the lanes road or not. In Fig. 2(c) we can see the vehicle in the center lane and between dashed lines. Applying the OpenCV Hough transform to detect the lines we can set the parameter which allows to find dashed lines. The length parameter sets the minimal length of the line. We have measured the maximal gaps in dashed lines and they are around 30 pixels.

The Hough transform result is represented in (5), where L is the lines set, r is the number of elements and L_x is the x line placed in the initial point p_o and in the end point p_f.

$$L = \{L_0(p_o, p_f), L_1(p_o, p_f), \dots, L_{r-1}(p_o, p_f), L_r(p_o, p_f)\} \quad (5)$$

Then, an unsupervised classifier has been used to find the optimal lines in the set L. Here, we defined initial conditions for each parameter in the classes. There are two classes in the classifier, Class 1 for the left line and Class 2 for the right line, defined by $C1$ and $C2$ respectively.

There are two parameters for each class. The first one is the middle point of the line and the second one is the slope of the line. The initial conditions of these parameters are defined in (6), (7), (8) and (9).

$$Pmc1 = (x_o, y_o) = (30\% * I_{x,max}, 75\% * I_{y,max}) \quad (6)$$
$$Pmc2 = (x_o, y_o) = (70\% * I_{x,max}, 75\% * I_{y,max}) \quad (7)$$
$$Mc1 = -I_{y,max}/I_{x,max} \quad (8)$$
$$Mc2 = I_{y,max}/I_{x,max} \quad (9)$$

Where, $Pmcx$ is the middle point of the class x, x can be the first class or the second one, $I_{x,max}$ is the numbers of the columns in the image, $I_{y,max}$ is the number of rows and Mcx is the slope of the class x line. The scaled values in (6) and (7) have been obtained by average from 20 pictures.

The two classes, $C1$ and C2, with their parameters are defined in (10) and (11) respectively.

$$C1(Pmc1, Mc1) \quad (10)$$
$$C2(Pmc2, Mc2) \quad (11)$$

The parameters change for each frame to adjust to the real lines on the road. We used an adaptive method for this deal which kept closer the class lines parameters to the real lines. If l_P line satisfies (12) it will be classified in Cx class, if not, line will be discarded.

$$l_P \rightarrow C_x \ \ if \ \{d_{P.Pmcx} < d_{minPx} \ and \ d_{PMcx} < d_{minMx} \quad (12)$$

Where, $d_{P.Pmcx}$ is the euclidean distance from the middle l_P point to $Pmcx$ point in the Cx class, $d_{P.Mcx}$ is the euclidean distance from the slope l_P line to Mcx slope in Cx class, d_{minPx} is the minimal distance between middle l_P point and $Pmcx$ and d_{minMx} is the minimal distance between slopes. The minimal distance d_{minMx} is a constant whose value is 1.5. The minimal points distance is a dynamic value and it starts with an 85% of the diagonal. Both minimal values have been defined by heuristic methods. The second value decreases easily in the next frames.

The $Pmcx$ parameters are updated in each new image. In (13) we describe this method.

$$Pmcx_{k+1} = \#l^{-1} \sum_{i=1}^{\#l} [(1+\alpha)Pmcx_k + (1-\alpha)PmL_i]/2 \quad (13)$$

Where, α is the decision factor, $Pmcx_k$ is the middle point in Cx class for actual k image, PmL_i is the parameter middle point for the i line classified, $Pmcx_{k+1}$ is the middle point for the next $k+1$ image and $\#\, l$ is the number of lines clasified in the Cx class.

The α factor is in the (0,1) range and it is defined in (14). This is a weight which depends on the $D_{PmLi-Pmcx_k}$ distance between the middle point from the line in study and the class point. If this distance is short the weight of this point will be higher in the next parameter class. Otherwise, if the point is very far its contribution will be lower.
This factor is based on a parabolic function.

$$\alpha = 1 - (D_{PmLi-Pmcx_k}/10)^2 \quad (14)$$

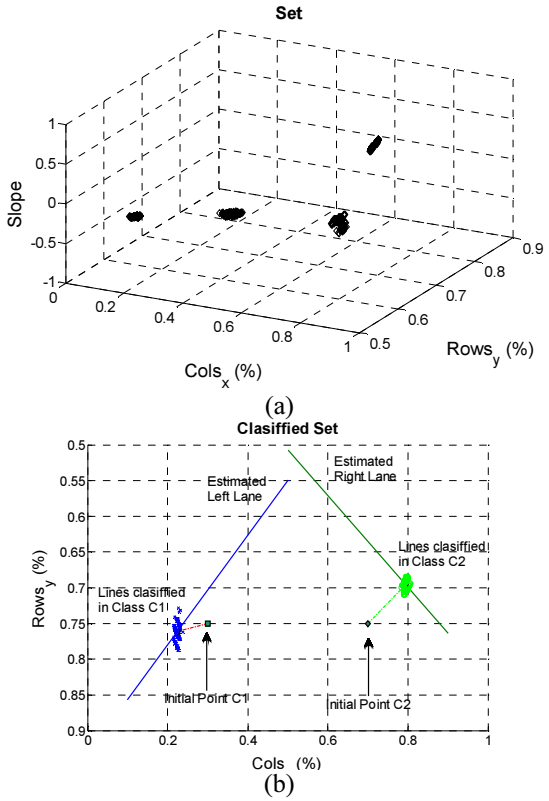

Figure 3. Lines classifier. (a) Input set. (b) Lines classified.

In Fig. 3 (a) we show a 3D input data set. In x axes are the columns, in y axes are the rows and in the z axes is the slope. The classified data is showed in Fig. 3 (b), where the lines blue and green are the classes $C1$ and C2 respectively and it is also showed the elements classified in each class. We can see the initial middle points and its path to the final place.

(a) (b) (c)

Figure 4. Lines Classifier. (a) Initial values of parameters, First frame. (b) Second frame. (c) Fourth frame.

Here, the $Pmcx$ parameters for each class and their adjust to the real lines are visible.

Although in the Fig. 3 (a) the input set can seem very easy to classify, we have followed a previous steps to obtain this simple data set. For this example we stored a data set of lines from images in a real road but are not the same images in Fig. 1 or Fig. 2.

Figure 4 shows the first frame and the lanes road detection. In 4 (a) it is showed the first frame in which we can see the initial conditions of parameters and the lines which represents the classes. The $C1$ class is represented on the left side and the $C2$ class is on the right side, both in blue color and their middle points in red. The green lines are the ones classified in the classes $C1$ or $C2$. The middle point for each line is also painted. In the border of the image there are some middle points of lines which have not been classified in any class.

In the second frame, Fig. 4(b), the parameters are closer to real lanes and in the fourth frame the estimated lines are almost over the real lanes. In the next frames the lines will be closer to real lanes.

However, in this paper we do not solve the curve lines problem, although our work can find the estimated lanes road in this cases too, but without curves.

The Tensor described in (15) holds the lines in their middle points with D_k . This is the distance between points like (16) describes. The Tensor avoids that the estimated road increases its width. This can happen if there is a lane derivation to the right side of the road and appears a new line on the road. The Tensor T_k is defined by the points $P1$ and $P2$ in (15). The middle points $Pmc\boldsymbol{x}_k$ of each line are validated by ΔD, which is the derivate of the distance in time and g is the minimal ΔD that it may change (17).

$$T_k = (P1(x,y), P2(x,y)) = (Pmc1_k, Pmc2_k) \quad (15)$$

$$D_k = \bar{d}(P1, P2) \quad (16)$$

$$Pmc\boldsymbol{x}_k = \begin{cases} Pmc\boldsymbol{x}_k & \Delta D < g \\ Pmc\boldsymbol{x}_{k-1} & \Delta D \geq g \end{cases} \quad (17)$$

Finally, in this phase it is applied the well known Kalman filter, whose design is described in a previous work presented in [11] an based on [12]. This filter is applied to the classifier parameters, these are: the middle points and the slopes in each class.

C. *Phase3. Vehicle position estimation and highlighted lanes.*

In this phase we obtain the estimated position of the vehicle on the road and, it is highlighted the lanes of the road in the image to help the user.

For the first step we use a trigonometric method. As we had described, we do not use the Euler angles to place the camera into the vehicle, so, the estimated position is relative and it is not given in meters but in percentage like a visual image in the image with warnings if the vehicle is on the edge of the road.

The angle formed by the two lines of the estimated road is bisected. The bisector cut the lower line of the image in the Pv point. Then, with (18), we found the difference ΔP between Pv and the middle point Pi in x axes in the $I(i,j)$ image.

Figure 5 shows the described process. Here, the bisector is in black and it cuts the x axes in Pv point. The lines road are in blue.

$$\Delta P = Pi - Pv \quad (18)$$

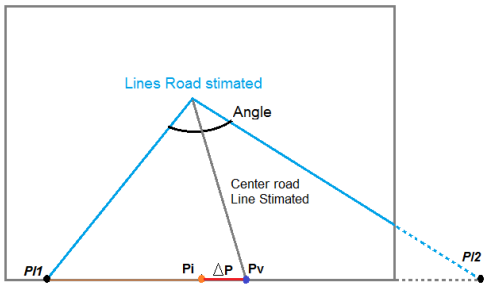

Figure 5. Vehicle position estimation.

If the Pv point is higher than point Pi in x axes, like in Fig.5, the ΔP difference will be negative. So, a negative ΔP means that the vehicle is on the left side of the lane road. It has been measured the distance from Pi to Pl1 to know how far the vehicle is to the center of the road. The Pl1 or Pl2 points can be out of the image, in this case it is necessary to find it out of the image, like in Fig. 5 the Pl2 point.

The second step in this phase is to highlight the real road lanes in the image. For this, we built a binary image $M(i,j)$ in black, in which are placed the estimated lines of the road in sticky white. Then, we applied an **AND** logic operator between the binary image $M(i,j)$ and the filtered image

$I(i,j)$. The $N(i,j)$ image is the results of this last process and it is described in (19).

$$N(i,j) = M(i,j)\boldsymbol{AND}\ I(i,j) \tag{19}$$

The pixels in $N(i,j)$ image have logic values. Then, these pixels are placed in the input image as it is described in (20), where, $fr(i,j)$ is the highlighted image.

$$fr(i,j) = \begin{cases} f(i,j) & N(i,j) = 0 \\ \boldsymbol{color} & N(i,j) = 1 \end{cases} \tag{20}$$

The parameter $\boldsymbol{color}$ in (20) is the RGB value to highlight the lane. We chose RGB (255,0,0), which is a full red color.

The user can see the real highlighted road lanes in red and the lanes estimated in blue with their center points in red, as we show in Fig. 6.

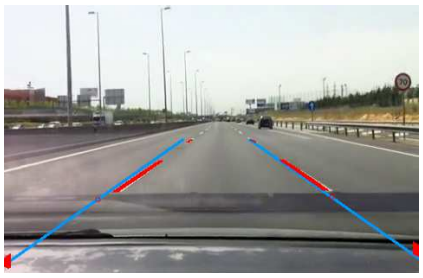

Figure 6. Lanes Highlighted in red.

III. EXPERIMETAL RESULTS

We tested the proposed algorithm in 5 videos, each one in a different environmental condition. The first is in the morning, the second in the middle day, the third in the afternoon, the forth at night and the last video is in a busy road. We show the detection results in Table 1.

TABLE I. EXPERIMETALS RESULTS IN DIFERENT ENVIROMENTAL CONDITIONS

Video	*Condition*	*Trues frames (%)*	*detection ready (frame)*
1	Morning	88	10
2	Middle day	90	7
3	Afternoon	82	11
4	Night	89	6
5	Traffic	85	10

These results show that the algorithm work better in middle light with the best light conditions. At night conditions there is also a good response. These results are good because we have prepared the image in phase 1 and the work in the next phases is easier. The worst detection is in the afternoon video, this is because in this time of day the sun light can hit the camera directly, and there are some problems to detect the lanes. If the conditions are good then the road detection will be ready in a few frames like in video 2 or 3. However, the algorithm is slow to detect the lanes in traffic conditions. If the detection fails, the process is restarted. Figure 7 shows the night video and the lanes road detection. The green rows indicate the vehicle deviation and the user should correct the position to left. If the position is further the arrows will change to red color to warn the user. The number of arrows depends on the deviation.

IV. CONCLUSIONS AND FUTURE WORKS

In this paper we proposed a new approach to detect the lanes road based on a monocular camera, Hough transform and an unsupervised classifier.

First of all, we have used a mono camera and our algorithm can be implemented in smart-phones or things like this.

We used three phases to obtain the true lanes. This algorithm has been tested in different videos and the results are good, as we show in Table 1, Fig. 6 and Fig. 7.

In phase 1 we used the brightness of image to know the environmental conditions. It is easier to find the lane marks using this method because the marks pixels have more brightness than the other ones.

Also, we improved a filter to detect the color lanes.

Figure 7. Night test. Image after phase 3.

The unsupervised classifier, in phase 2, allows finding the best lines to build the estimated road.

An unsupervised classifier is very useful when the elements to classify and the parameters of the classes are changing.

The Kalman filter reduces the gaussian noise from an estimated road and it allows tracking the lane.

The lanes highlighted and the vehicle positions help to the driver to be in the center of the road. If the driver could not see the image, he can hear a warning sound if the vehicle is on the road edge, but it will be implemented in next works.

Also, in future works, we propose to improve the detection in lane changes. We have started to reduce the algorithm and we will work in an interface to the user in order to make an application for Android SO.

Our entire algorithm has been implemented in OpenCv.

REFERENCES

[1] A. B. Hillel, R. Lerner, D. Levi, y G. Raz, «Recent progress in road and lane detection: a survey», *Mach. Vis. Appl.*, pp. 1-19.

[2] C. Guo y S. Mita, «Drivable road region detection based on homography estimation with road appearance and driving state models», en *4th International Conference on Autonomous Robots and Agents, 2009. ICARA 2009*, 2009, pp. 204-209.

[3] J. M. Alvarez, T. Gevers, y A. M. Lopez, «3D Scene priors for road detection», en *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 57-64.

[4] J. M. A. Alvarez, T. Gevers, y A. M. Lopez, «Vision-based road detection using road models», en *2009 16th IEEE International Conference on Image Processing (ICIP)*, 2009, pp. 2073-2076.

[5] J.-W. Kim, T.-H. Kim, y K.-H. Jo, «Traffic road line detection based on the vanishing point and contour information», en *2011 Proceedings of SICE Annual Conference (SICE)*, 2011, pp. 495-498.

[6] E. R. M. Faizal y H. M. A. H. Mansor, «Virtual Mid-Line Detection on Curve Road for User Guidance Using Simulation Model», en *International Conference on Computer Technology and Development, 2009. ICCTD '09*, 2009, vol. 1, pp. 24-27.

[7] G. Mastorakis y E. R. Davies, «Improved line detection algorithm for locating road lane markings», *Electron. Lett.*, vol. 47, n.º 3, pp. 183-184, 2011.

[8] D. Fontanelli, M. Cappelletti, y D. Macii, «A RANSAC-based fast road line detection algorithm for high-speed wheeled vehicles», en *2011 IEEE Instrumentation and Measurement Technology Conference (I2MTC)*, 2011, pp. 1-6.

[9] F. M. Sebdani y H. Pourghassem, «A robust and real-time road line extraction algorithm using hough transform in intelligent transportation system application», en *2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE)*, 2012, vol. 3, pp. 256-260.

[10] M. Nieto, J. A. Laborda, y L. Salgado, «Road environment modeling using robust perspective analysis and recursive Bayesian segmentation», *Mach. Vis. Appl.*, vol. 22, n.º 6, pp. 927-945, nov. 2011.

[11] A. Cela, J. Yebes, R. Arroyo, L. Bergasa, R. Barea, y E. López, «Complete Low-Cost Implementation of a Teleoperated Control System for a Humanoid Robot», *Sensors*, vol. 13, n.º 2, pp. 1385-1401, ene. 2013.

[12] R. Kalman, «A New Approach to Linear Filtering and Prediction Problems», *Trans. Asme – J. Basic Eng.*, n.º 82 (Series D), pp. 35-45, 1960.