Universidad del País Vasco Euskal Herriko Unibertsitatea

Unibertsitate Masterra

Konputazio **I**ngeniaritza eta **S**istema **A**dimentsuak

Konputazio Zientziak eta Adimen Artifiziala Saila –
Departamento de Ciencias de la Computación e Inteligencia Artificial

K I S A

I C S I

Master Tesia

Guided Bus Rapid Transit

Unai Elordi Hidalgo

Tutorea(k)

Carmen Hernández Gómez

Konputazio Zientziak eta Adimen Artifiziala sailaInformatika Fakultatea
Informatika Fakultatea

Master eta Doktorego Eskola
Escuela de Máster y Doctorado
Master and Doctoral School

KZAA /CCIA

2016ko iraila

Master eta Doktorego Eskola
Escuela de Máster y Doctorado
Master and Doctoral School

GUIDED BUS RAPID TRANSIT

Unai Elordi Hidalgo

September 2016

Contents

List of Figures

List of Tables

Abstract

Electrical Bus Rapid Transit (eBRT) is a charging electrical public transport which brings a clean, high performance, and affordable cost alternative from the conventional traffic vehicles which work with combustion and hybrid technology. These buses charge the battery in every bus stop to arrive at the next station.

But, this charging system needs an appropriate infrastructure called *pantograph*, and it requires a high precision bus location to maintain battery lifetime, energy saving and charging time. To overcome this issue Vicomtech and Datik has planned a project based on computer vision to help to the driver to locate the vehicle in the correct place. In this document, we present a mono camera bus driver guided fast algorithm because these vehicles embedded computers do not support high computation and precision operations. In addition to the frequent lane sign, there are more accurate geometric beacons painted on the road to bring metric information to the vision system. This method uses segmentation to binarize the image discriminating the background space. Besides it detects, tracks and counts different lane mark contours in addition to classify each special painted mark. Besides it does not need any calibration task to calculate longitudinal and cross distances because we know the lane mark sizes.

1 Introduction

This document describes an electrical Bus Rapid Transit lane marking detection system development which is targeted to be a commercial product for Datik. This project is a collaboration between two centres. On the one hand, Vicomtech is an Applied Research Centre specialized on visual computing and multimedia technologies. On the contrary, Datik is a startup member of Irizar Group, which researches and innovates on Bus embedded systems.

Electrical Bus Rapid Transit (eBRT) is a flexible, high performance and rapid transit transport. The evolution of buses can be traced as combustion engines to hybrid technology towards to pure electricity buses. Nowadays, they are already riding worldwide with different technical solutions.

They can travel anywhere that there is a pavement, and the cost of infrastructure is relatively small comparing to the train or subway infrastructure.

The main feature of eBRT is the capacity of fast charging. It charges the battery in each bus stop, with the objective of preserve battery autonomy. The charging average time is about 20 seconds.

A low-emission and energy efficient public transport network demand an appropriate charging infrastructure which is called *pantograph*. The top-down pantograph is a fast-charging system that can mount on a mast or roof of a bus stop, see figure 1.

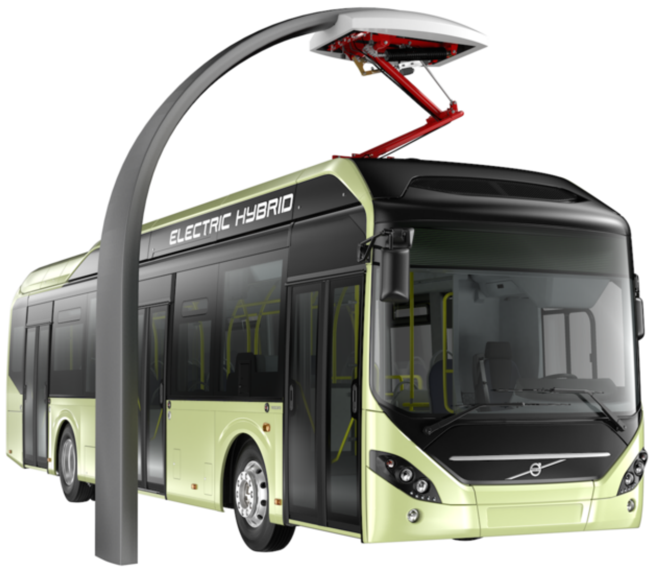


Figure 1: Pantograph on the roof

This flexible off-board charging solution is available from multiple bus types. The idea behind the pantograph is to charge the batteries sufficiently, for example at a terminus, so that the buses have enough power to travel to the next charging bus station. For fully automatic charging, the electrical bus is driven under the charging station, which then lowers the pantograph onto the contact rails. It works with Wireless communication via Wi-Fi IEE 802.1. The below representation is an example of Bus Rapid Transit station, see figure 2.

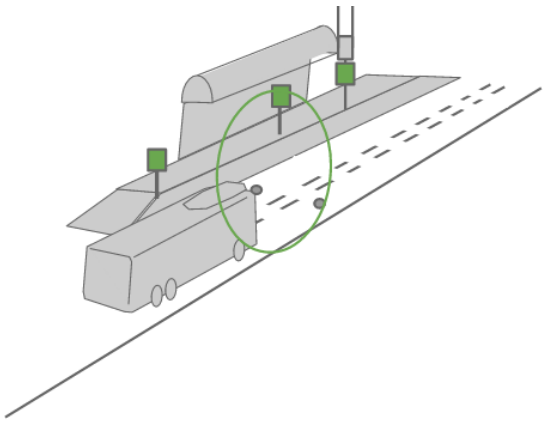


Figure 2: eBRT bus stop

As it can see in the image 2, there are painted lane markings along the bus shelter where the charging system is located.

Unfortunately, pantograph charging has some metrical restrictions to ensure that the charging system works correctly. The driver must leave the bus correctly, in the place where the pantograph is located with +-2 cm error. In this way the coachman has to solve two issues.

Firstly, the operator has to estimate the distance and the optimum speed until the bus station final mark line. Besides, there are beacons installed on the road which they are designed to notify to the driver remaining longitudinal distance information in meters.

The second task is more difficult. Although there are painted landmarks which define lane centre and pantograph location, the driver has no any metric information to ensure where is precisely located and therefore, it depends on his skills and his visual estimation abilities. The figure 3 shows a bus driving on a lane markings road.

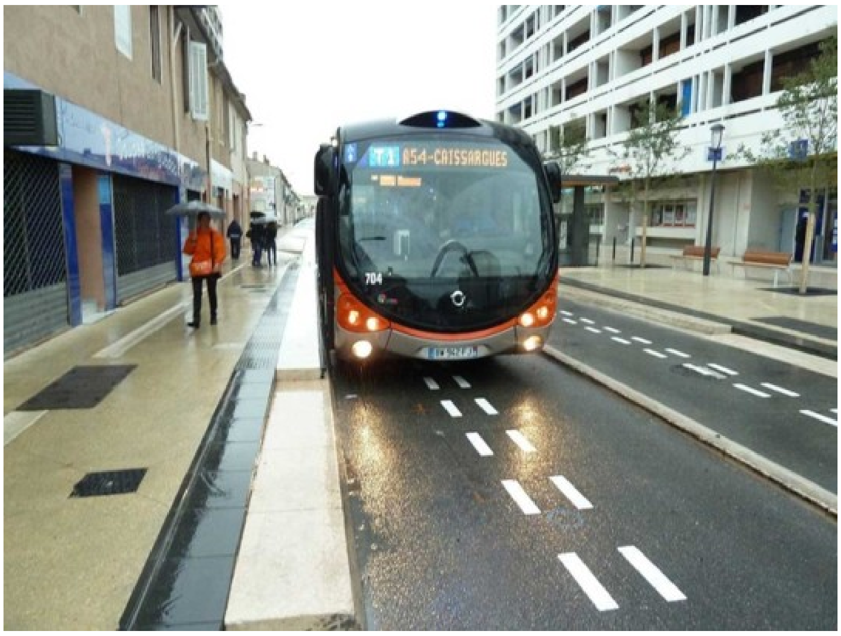


Figure 3: eBRT riding with lane marking system

So, the beacons do not solve the effectiveness of bus stop process, because there is some uncertainty which depends on the driver skills. The chauffeur should have several metrical references to make more efficient the bus stop process.

Recent advances in computer vision systems on vehicles provides a low-cost solution to address motion estimation, tracking and road marking detections.

1.1 Motivation and Objectives

The main idea of this master thesis is to develop a guided bus system to assist the driver. The algorithm helps to locate the bus in the road defined marks accurately, with high precision, exactly below the pantograph.

As it is mentioned before, the pantograph charging infrastructure has critical device restrictions, because this error affects straightaway on charging efficiency, battery lifetime, autonomy and saving energy. So, Computer Vision algorithms allow solving these troubles.

ADAS (Advanced Drive Assistance Systems) has been widely researched for many years, and its applications are nowadays very helpful for safety driving of many scenarios on vehicles which one of them could be the eBRT guided system.

Talking about ADAS systems, for example in [8] and [25] propose Driver Fatigue Warning. In those research, face tracking is used to recognize fatigue yawing situations. It uses the information about skin colour, face location, eye tracking and mouth location.

Other scenarios of ADAS systems work in the area of vehicle localization, and they are looking to the road. Most of them focus lane tracking detection LDW (Lane Departure Warning) and collision avoidance system (CAS).

Both methods need lane markings as a reference from the road. Besides, this beacon metrics are standard measures, and it allows to establish a connection between real metric information in centimetres and pixel information. This project describes a new motion estimation algorithm where it is based on special lane markings as it can be seen in the figure 4.

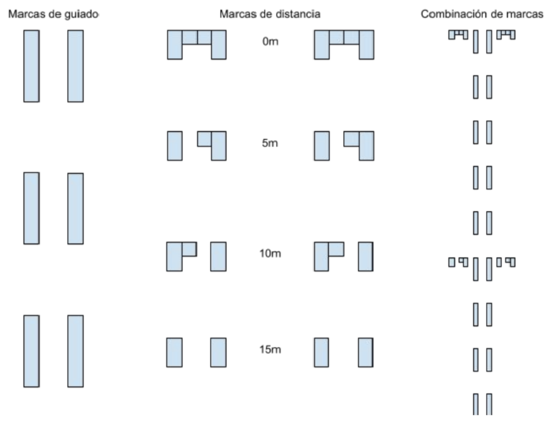


Figure 4: From left to right: guide lane marking, distance lane marking (15m, 10m, 5m, final-line), lane mark design in the road.

There are two different groups of lane markings, distance markings and guide markings. There are four geometric figures to define distance markings, which begins on 15 meters and finishes in the final-line mark. In addition, every geometric mark is replicated on each side to improve the safety detection. The second type is targeted to help the driver, maintaining the bus centred, as much as possible on the lane. In other words, they are designed as a reference which allows to computer vision algorithm, where is located respect to road centre in centimetres.

This lane marks eBUS control system implies to create image processing algorithm which divides several steps. The block diagram below shows how the algorithm works.

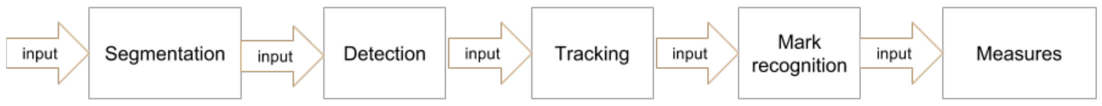


Figure 5: Process algorithm diagram.

Firstly, the image has to be segmented because the algorithm needs to focus on the part of the picture which is relevant for the next steps. Besides thresholding methods help to segmentation process for binarize images.In conclusion, the information which is not relevant for our algorithm, and besides, this binarization can provide better performance for the next steps.

The next task is the detection step, which its input, is the segmentation output image. The process searches contours on the picture where they could be significant for the algorithm taking into account some constraints which they are going to explain in section.

Later, this frame per frame detection should be able to recognize contours that are tracked in time t and $t+1$. The tracking methods allow measuring how much has been displaced in pixels through time in the horizontal and vertical distance because each coordinate is assigned for each lane marking. These coordinates are not in the projective space, that is why it is necessary to create a relation between the image position with the road plane, such as a rotation and translation matrix which is called homography. Finally, this picture tracking points will be projected to road plane.

The last process before to end, it is the classification task. It aims to classify different distance markings as mentioned earlier. There are four different geometric group of figures, and there are located to define distance references until the bus final stop marking which is the 0-meter figure.

Finally, the algorithm gets output information taking into account homographic data. But, thanks to the fact that lane marking sizes are fixed it avoids scale factor problem because it is able auto calibrate pixel and metric data relation.

In addition to making a robust algorithm, it must be fast enough to execute in bus embedded systems. These devices are not as powerful as the traditional computers. Firstly the processors are so much slower, and frequently they are arm systems. Besides, they have not contain powerful graphic cards, where it would allow to make graphic based processing (GPU processing). Therefore the computation constraints are very high. The iteration speed can determine the way to choose different algorithms, and maybe there will be algorithms which are going to modify according to the domain space.

1.2 Outline

The document is divided into many sections. Firstly an abstract and introduction are written with the aim of motivation and objectives. After, a related work section talks about the lane detection, tracking methods, marker classifications, or many other researching methods until now.

The Approach section describes what is the methodology to resolve the asked requisites. Experimental results section shows the behavior of algorithm comparing to other algorithms. Finally, the Conclusions represent the development project results and make a feedback about the final result.

2 Related Work

As it is mentioned before, the goal of this project is to create an automatic guided system of eBRT based on Computer Vision algorithm which its input are digital images. Those can see as two-dimensional matrices where each cell represents a pixel, and the colour depends on the number of channels such as grayscale (1 channel) or RGB images (3 channels).

This colour intensity or contrast data is very helpful for computer vision techniques. For example, the images with high contrast values have more feature key points which bring valuable information to the algorithm input.

Fortunately, our approach input images include high contrast areas, because the camera is looking to the road and the horizon, where there are white lane markings painted on the black road. Besides, there are user defined markings which are very helpful to get information such as longitudinal distance, traversal distance and relative localization in the road lane.

Lane marking detection and tracking involves many computer vision transformations. Mainly the input image is captured by a camera and transformed to grayscale. Later it is necessary to remove image noise and get relevant information. After this, lane marking contours are detected in each binarized frame, and finally each detection boundaries will be tracked to measure eBRT metric information. In few words, there are five different issues to solve: segmentation, lane marking detection, lane marking tracking, recognition and projective issues.

2.1 Segmentation

In computer vision, the concept of segmentation is widely used in almost many computer vision transformations. Its main task is to separate the interesting part from non-interesting part (noise, background, etc.). However, in this process, some information is lost, and that is why there are many methods which uses segmentation with different missions, such as face detection, road segmentation, etc.

The first step is usually the pre-processing or thresholding algorithms. In addition to remove noise, this method also eliminates the texture, because can disturb contour boundary detection.

In segmentation, in addition to thresholding, there are different ways to overcome noise trouble, for example, local operators. It can classify as a linear and non-linear methods.

Finally, contour detection methods are analyzed which they are divided into two main groups called global operators.

2.1.1 Thresholding

In image processing, thresholding technique is well-known on image segmentation area. It is the most basic operation to binarize the image to separate the Region of Interest (ROI) from the background.

The simplest way to apply thresholding is to use a static threshold. In few words, if we have an image called $I(x, y)$ where x and y are the coordinates of the picture, the result goes between 0 and 255 depending on grayscale value of I(x,y). If it is defined a static threshold called th, this equation will be true.

$$I(x,y) = \begin{cases} 0 & if \;\; f(x,y) < t \\ 255 & if \;\; f(x,y) >= t \end{cases} \tag{1}$$

Static thresholding doesn't work fine in many digital images due to its static parameter t, and it is not able to adapt his threshold for different digital images.

About adaptative thresholding, [22] presents a multiscale non-parametric and unsupervised method of automatic threshold selection which is commonly called Otsu thresholding. In this work, it evaluates the goodness of threshold value through automatic thresholding. The image divides two different classes, for example, background and foreground objects. The algorithm uses gray level histograms, based on zero and first-order cumulative moment. Unfortunately, the selected threshold become less credible as long as class numbers increase. Otsu's method shows good performance on bimodal distributions but is not suitable to multimodal distributions.

Other typical preprocessing method is the bilateral filter or mean shift filter [23] or [7]. On the one hand, a non-iterative scheme is presented whereby means of a non-linear value combination of neighborhood images, looking at photogrammetric values. The principal idea is to compute local weighted average gray values in the neighborhood. In the case of bilateral filter, decreasing function is defined like $d = |I(p) - I(r)|$ and $r \in N(r)$ where r is a point of the neighborhood. On the other hand, The weights of mean shift filter define the local density distribution of vector $z = |p, I(p)^T|$.

In addition to these non-linear filters, we can use also Rank Order Filters where the difference with mean shift or bilateral filter is the neighborhood which defines a rank of $i = 1, \ldots, N$. The order starts from the darkest level to brightest level.

One of the most important contribution of this project is to find a robust and the fast way to detect lane boundaries which the edge segmentation algorithms are very helpful. In addition to thresholding operations, the local operators bring edge detection alternatives which are discussed in following section.

2.1.2 Local operators

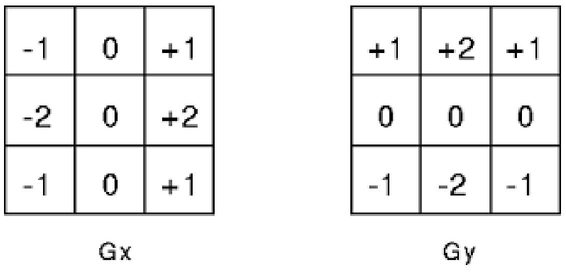
There are many methods to use local operators image transform, but mainly there are two types which are called differential and phase congruence methods. The first is based on Convolution between two functions, $f(I)$ and $g(I)$ where $f(I)$ is a function that assigns a intensity value to each pixel of the image, $g(I)$ is quadratic matrix called kernel function where each pixel is calculated by multiplying the kernel value by the pixel value corresponding to input image, and finally, the output image is going to be transformed to edge resulting frame as the equation below.

$$output = f(x) * g(x) \tag{2}$$

This operation repeats the process for every pixel in the image changing his intensity.

The Convolution Kernel is a fixed matrix of numerical coefficients and can be seen as a template which the output is expected to be high on several types of edges, and low on flat or uniformly textured areas. The size of the kernel is called *support*, and the selection kernel size changes the image output. Large kernels have a better approximation, but the small ones are very sensitive to noise.

Using convolution in edge detection, Sobel [29] operator has been a good choice for scientific researchers. The method is a derivative approximation. In fact, it is not a derivative at all. It is an approximation of discrete space being a local fit of a parabolic function. A Sobel filter has two kernels, x-direction kernel and y-direction kernel. The x-direction kernel detects horizontal lines, and the y-direction kernel detects vertical lines. The figure 6 shows kernel implementations of both axes.



-1	0	+1
-2	0	+2
-1	0	+1

Gx

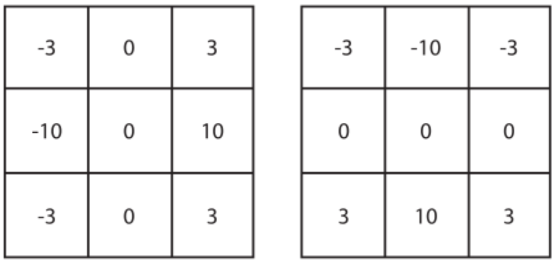
+1	+2	+1
0	0	0
-1	-2	-1

Gy

Figure 6: Sobel operators

Gx is the x-order kernel and *G*y is the y-order detector. Although Sobel derivative brings a fast edge detection operator, it is sensitive to noise in case of small kernels and it is slower as long as the kernel is getting larger.

Moreover, Sobel filter handles x and y-axis derivatives, but is poorly designed to handle directional derivatives. However, there is an alternative called Scharr operator [28] which overcomes the problem, giving an accurate output with 3x3 kernel.



-3	0	3
-10	0	10
-3	0	3

-3	-10	-3
0	0	0
3	10	3

Figure 7: Scharr kernel

Sobel operator is not the unique method in the spatial domain, for example in [16], there is a Laplace operator applied for computer vision. This method is often used for edge detection or motion estimation. The discrete Laplacian is defined as the sum of the second derivatives in x and y-axis, as it is shown in the equation below.

$$Laplace(f) \equiv \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{3}$$

Since the input image is a representation of discrete intensity pixels, the method needs to obtain a convolution kernel so it can approximate the second derivatives in the definition of the Laplacian. The most used kernels can be seen in table 1.

Table 1: Laplace kernels

(a)

0	-1	0
-1	4	-1
0	-1	0

(b)

-1	-1	-1
-1	8	-1
-1	-1	-1

The main application of Laplacian in computer vision is blob detection. The Laplacian operator is a weighted sum of second derivatives along x and y-axis. So, that means that a single point or any small blob that is surrounded by higher values will tend to maximize this function. On the other hand, a point or small blob that is surrounded by lower values will tend to maximize the negative of this feature.

In contrast to Laplacian operator, Canny operator [5] works as follows:

- The first derivatives are computed.
- Later they combine to four directional derivatives.
- The points where these directional derivatives find a local maximum are candidates to be and edge.

This method proposes to optimize the template filter, which the edge detection is very close to the convolution of the input with the first derivatives of a Gaussian function.

As the algorithm is more precise than the others explained before, the algorithm requires to process many times the convolution. However, with the current computers, this performance is not perceptible for the users.

In addition to differential operators, phase congruence (PC) methods has been researched throughout the years. Human psychological perception on images demonstrates that the human vision has a good response to images which there are in phase and highly ordered [21], [20], [19]. Originally these methods are based on local energy definition, and the most useful filters are called Gabor quadrature filters.Its main idea is to maintain the equilibrium of spatial and frequency indeterminacy. In contrast to linear filtering based on local operators, phase congruence methods can detect roof, peak and step edges. Moreover, these edge detectors satisfy the idempotence condition [26].

Superpixel methods are becoming famous thanks to low computational complexity, simplicity and the required few parameters. For example [1] uses SLIC (Simple Linear Iterator Clustering) to segment the pixels in five-dimensional **rgbaxy** regions or clusters (Red, Green, Blue, X-position, Y-position) by reducing the computational complexity. The main idea is to create N number of pixels groups, where the size of them are relatively similar. It needs few parameters, and it is better than the other superpixel methods.

However, local gradient operators tend to be noisy by nature. For instance, high traffic, other vehicles on the road or shadows.

In [13] and [14], an approach of the new low-level descriptor has been published where its aim is to bring more reliable lane marking detections. In both cases, ridgeness defines a measure of how well resembles a ridge.

If the image contrast is well-contrasted, lane markings rigde can be obtain using a threshold value. A ridge can describe better lane boundaries detection thanks to values and orientations of ridge. The most brighter values are placed on lane marking centers and as long as it's been on marking, the edge values are going to decrease. Ridgeness can be formulate mathematically as follows:

1. Firstly, the input image $I(X)$ is convolved with 2D Gaussian $G_\sigma(X)$ filter with σ variance in this way, $L_{\sigma_d}(X) = G_\sigma(X) * I(X)$
2. Compute gradient vector, $W_{\sigma_d}(X) = (\partial_u L_{\sigma_d}(X), \partial_v L_{\sigma_d}(X))^t$.
3. Compute the structure tensor field, $S_{\sigma_d,\sigma_i}(X) = G_{\sigma_i}(X) * s_{\sigma_d}$ where $s_{\sigma_d} = W_{\sigma_d}(X) * W_{\sigma_d}(X)^t$
4. Take eigenvector of best eigenvalue, $P_{\sigma_d,\sigma_i}(X) = W'_{\sigma_d,\sigma_i}(X)^t * W_{\sigma_d}(X)$.
5. Define vector field, $\tilde{W}_{\sigma_d,\sigma_i}(X) = sign(P_{\sigma_d,\sigma_i}(X))W'_{\sigma_d,\sigma_i}(X)$
6. Finally, $\tilde{K}_{\sigma_d,\sigma_i}(X) = -div(\tilde{W}_{\sigma_d,\sigma_i}(X))$

2.2 Contour detection or global multiscale operator

There are 3 types in contextual analysis: contour saliency, grouping pixels and active contours.

2.2.1 Contour Saliency

There are psychophysical and neurophysiological studies [24], [17] which demonstrates that answer of human visual system to an oriented stimulus is influenced by the presence of stimuli in the surroundings, and two methods classified this fact:

- Facilitation from stimuli which are collinear with central one.
- Surround suppression or inhibition by other stimuli.

The paper [9] proposes a simple inhibition term T which defines local average on a ring around each pixel. T is small in isolated edges and is high on texture surfaces. So it removes the texture and improves the visibility of contour objects. The surround suppression applies a post-processing step in edge detection. The best benefit of this method is that removes the texture noise where boundaries are not well defined.

The facilitation method is based on tensor voting. There is a vector called $E_x(r)$ oriented on the curve connecting r point by x. The magnitude of E is the value of such a likelihood where usually it is a decreasing function. After, each pixel is defined by the distribution of the values from all other pixels of the image.

The Relaxation Labeling method is a statistical approach of probabilistic graph based on contour detection successfully developed on [3] [27]. Actually, in its simplest form, there is a node group S which is interconnected by labels L of graph G. In other words, the graph called G has a set of nodes $S = s_1, \ldots, s_N$. These nodes connect the labels which are covered by compatibility restrictions R. So, a labeling L is admitted by R, if the pair $L(s_i), L(s_k)$ belongs to a given set $Lik\ Li \times Lk$ for every pair (si, sk) of nodes which are joined by an edge in the graph G. The disadvantage of this method is that edge graph binary formulation can be very restrictive where no labelling is allowed or too permissive where more than one tagging is allowed.

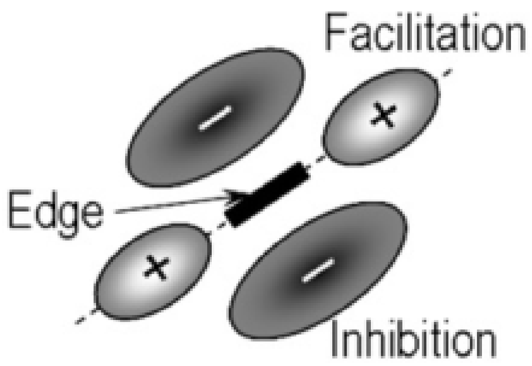


Figure 8: Facilitation and surround suppression

The article [12] introduces a probabilistic framework which implements graph theory with Markov Random Fields. There is a labelling p matrix which represents the degree of confidence of this label has been assigned to this object. So, this confidence values are interpreted as probabilities with average consistency map $a(p)$ where is proportional to Gibbs distribution.

The main problem of the relaxation is to find labelling p which maximizes $a(p)$ in the Markov Random Field framework. As maximization problem is an NP-hard problem, it is not possible to solve in polynomial time. In conclusion, to make the algorithm faster, it is necessary to select a suitable initial labelling and apply gradient descent iterative methods to solve the problem.

2.2.2 Grouping pixels according to Gestalt principles

According to Gestalt human vision perception principles, many near contours can be grouped as the same contour if those cover certain restrictions. These laws try to explain how the humanity can arrange some contours that apparently are separated such as similarity, good continuation, closure, symmetry.

There is a graph with nodes as pixels where they are connected each other by adjacent edges with a cost minimization function. Usually, this decreasing function takes into account collinearity and co-circularity. Although this graph theory is solved mathematically, it is not settled in computation time. There has proposed different cost functions, such as, geometrical or probabilistic. Though there are many Gestalt laws, the most researched methods are closure and symmetry.

Closure: Graph-based grouping algorithms usually identify subgraph $Gprima$ with G which minimizes certain function with the constraint that G' is a cycle.

In addition to its simplest method, there are more complex approaches such as Markov based methods [15]. A transition matrix P is computed where P_i, j is the conditional probability that the path goes from i to j. Rather than create a probabilistic model of the edge distribution, a simple cost function is minimized, based on the normalized length of the gaps between contour fragment and on the total curvature of the detected boundary. This cost function tends to smooth edges.

Symmetry: This law seeks a group of edge pixels which defines some forms where they can estimate some local or global symmetry in a curved balance edge pixels. This method introduces a new skeleton concept to define a shape in $2D$ space which commonly defines a central axis of m of S shape.

In [30] a new cost function is introduced as the ratio between two terms:

- The numerator measures the symmetry of sought symmetric region R and the denominator is equal to R area.
- New grouping element is introduced to decide the detected symmetry structure, defined as a trapezoid with previously identified edge segment.

2.2.3 Active contours

Active contours which are also called snake shape models [11], they are used in segmentation, object tracking, edge detection and stereo matching. These deformable models target is to evolve the curve C drawn manually by the user around an object through energy minimization. This minimization is a sum of two type of energies: internal energy E_{int} which measure the smoothness of curve C and E_{ext} external energy, where measures how close is curve from the user defined contour.

$$\begin{aligned} E_{int} &= \oint_c (\alpha||\dot{r}|| + \beta||\ddot{r})dl \\ E_{ext} &= \oint_c g(||\nabla I||)dl \end{aligned} \tag{4}$$

α and β are input parameters and g is a decreasing function where $g(\xi)$ will be zero as well as ξ tends to infinity.

On the one hand, internal energy terms are related to the length of the C curve and on the contrary, external power is little when the curve is located on the maximum of gradient. The figure describes the C curve evolution

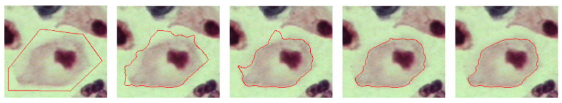


Figure 9: Initial C curve and the results after n=4,8,12,16 iterations

Snake algorithms applied to edge detectors brings better performance than local edge detectors especially on continuity constraints, smoothness and closure of detected contours. Moreover, it can identify illusory contours. However, unfortunately, initial conditions, high dependency, slow convergence and little capacity to recognize low contrast shapes make researchers conditional on choosing other techniques.

These algorithms can be classified in two types, *parametric snakes* and *level set snakes*. The first one is expressed by parametric equation $r = r(s)$ where $s \in [0, 1]$. The variable s represents the ratio between arc length, $r(0)$ and $r(s)$ of the whole curve. Level set snakes, however represents the space with a function $\Psi(r)$ where $\Psi(r) = 0$ on C.

In conclusion, parametric snakes, unless the algorithm detects complicated split or special merge procedures, the curve C will preserve the topology during the iterative algorithm. Level set snakes, by contrast, have better performance on topology changing. Besides, it improves the behavior where the number of elements to detect are not known. However, the noise robustness decreases in case of over splitting cases.

In addition to boundary based contour detection, there are region based contours and shape based boundaries algorithms. The first one, based on image integrals, is not as sensitive as the aforementioned active contours and it is more robust in case of image degradations, but they are computationally more expensive because of integral computation. The second one is targeted to detect complex contours, and it adds a new term E_{shape} in energy equation which makes similarity constraints between training objects and C curve. This approach is divided into two steps, vectorization of all shapes and the estimation of a probability distribution of the training shapes. These improvements add more accuracy on shape detection but, its main limitation is the high dimensionality of shape space.

2.3 Geometric object classification

Most of the research ADAS (Advanced Driver Assistance System) features, are based on lane tracking but not in lane marking recognition. However, there is previous work about road sign classification. For example, the article [4] presents a uncalibrated lane marking detection and a recognition method. Besides it can recognize lane boundaries using linear parabolic model. Finally, it uses a cascade classifier to discriminate five types of markings: dashed, dashed-solid, solid-dashed, single solid, double solid, as the figure 10.

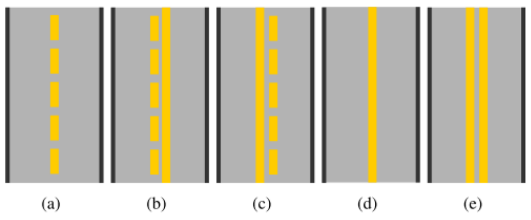


Figure 10: From left to right: (a) Dashed (b) Dashed-solid (c) Solid-dashed (d) Single solid (e) Double solid.

The first step segments the image using rigidness algorithm. After this ROI is defined on the image in order to reduce search space. Later, patches are defined inside of ROI with these sizes. In the equation below is defined patch size operation.

$$sizer_i = 0.03W * 0.2W \tag{5}$$

where W is the width of image. There are two types of patches ri and re in order to obtain binarized patches for classifications. Afterwards, the consistencies of these types are checked and then this binary image is obtained by this mathematical expression:

$$r_b^t(x,y) = \begin{cases} 1, \text{ if} \, r_e(x,y) \; > \; \mu + k\sigma \\ 0, \; otherwise \end{cases}$$

where μ and σ are the mean and standard deviation of r_i and k is the tolerance used to distinguish pavements and lane markings. This binary data will be used to discriminate in the classes mentioned before (dashed, dashed solid, etc). The classification is divided into three levels but actually four binary classifiers are created as it can see in the figure 11.

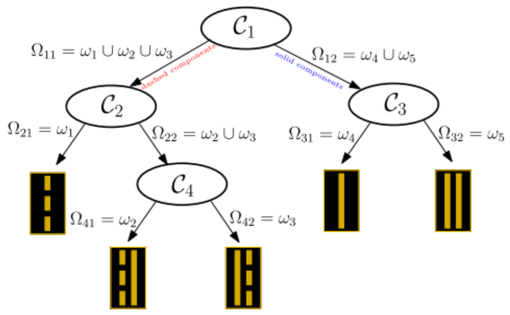


Figure 11: Schematic view of cascade classifier levels

Following classification task, the article [2] makes supervised classification on the road marking objects. This article classifies two general types of classes: dashed lane markers and continuous lane markers. Those classes provide an accurate description of an object to yield unique attributes. As it does not exist any public dataset to make the evaluation, they generated with those road marking signals.

The algorithm uses adaboost as a binary classifier. The binary decision trees with a tree depth of two levels used as a weak learner. The feature vectors used to define binary trees are based on grayscale level values between 0 and 1. Besides, it has a region of interest which is calculated by 256 bin histogram. So, the feature algorithm vector depends on width height of ROI.

About training, in the first cascade step, it begins with a set of feature vectors which the positive and negative classes are required. The positive value defines the road area and the negative holds non-road area. The feature vector computation uses the annotated label information to distinguish between two categories. After that, it tests each ROI looking for intersection with either road polygon. In conclusion, it follows the next scenarios:

- ROI is totally out of the road polygon.
- ROI is partially outside of the road polygon.
- ROI is in the road polygon and intersecting a car polygon.

The negative classes are considered if the road polygon is fully outside or if the car polygon is intersecting. If there is partially outside of road polygon, it is classified as a mixed class (negative and positive).

There are also standard template matching research papers in model traffic signs. For example, in [10], it shows the road sign detection and recognition is solved by Matching Pursuit method (MP).

Matching pursuit (MP) is a sparse approximation algorithm which finds the best matching projections of multidimensional data in a dictionary. It consists on an approximation of a signal f from Hilbert space H as a weighted sum of finitely many functions g (called atoms) taken from dictionary. In few words, the system has two phases, detection step and recognition step. Firstly, it takes the relative position of road sign using a priori knowledge, and it extracts the subregion of the traffic sign. Later this subregion extracts and compares with basic template matching. Then there is a part of recognition which is divided into training and testing. The training process consists finding the best matching on the set of the best MP filters which corresponds to different road signs. Thanks to the training step, the testing process projects the unknown input with a trained set of MP filters and it tries to find the best match.

3 Approach

This project aims to develop a vision based bus guided system helping the driver to make bus stop process fixed in top-down pantograph. The pantograph charging system is very restrictive because the accuracy depends on the bus station distance beacons and the driver skills. This action explicitly affects on bus fast battery charging and lifetime. So, the vision guided system will help to the driver to make more accurate bus stopping process.

The algorithm is only a module of the whole system, but to make robust and reusable project, Datik, the main project owner, defines requisites for algorithm output information.

- Recognize and classify distance marks.
- Measure the longitudinal distance in meters.
- Calculate the distance between camera centre and lane centre.

The road lanes are usually high contrast spaces making easier the segmentation task between white road figures and the black road. Besides, they are drawn especial lane marks which the algorithm recognizes as a reference to make computer vision operations.

These elements are geometric shapes which the algorithm will be able to identify a bus station approximation. Besides, the process uses the position of each mark to locate the bus in the projective space.

There are two types marks, localization marks and distance marks.

Localization marks are double-dashed essential marks with fixed size in the centre of the road to the bus stop like the figure below.

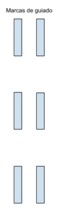


Figure 12: Localization markings

Double-dashed marks are symmetric and each mark sizes are 10 cm width and 50 cm height. The longitudinal distance between a mark and the next mark is one meter.

Distance marks instead, has two different meanings. On the one hand, each geometric mark indicates the remaining distance to the end of the bus stop which the beginning mark is 15 meters, and the end mark is 0 meters. Furthermore the order of the marks defines the direction of the bus station because there may be bus stops on the both sides and the algorithm should not be confused in the left lane markings.

So, the coach stop road is painted with the combination of both types as it can see in the figure 13.

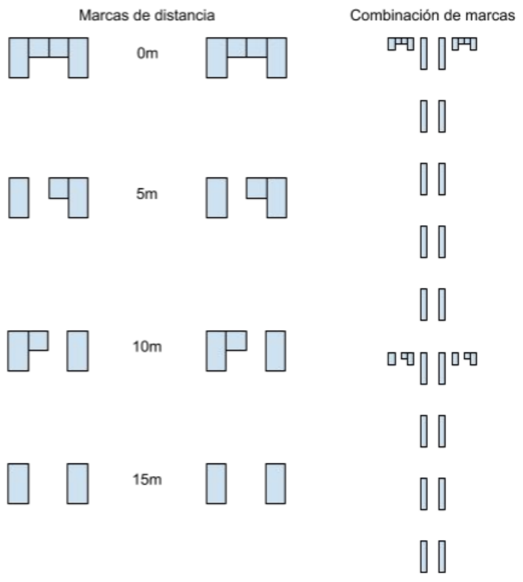


Figure 13: Special distance markers (left) and combination of distance and localization marks (right).

Although the related work talks widely about contour segmentation and detection, most of these research papers are not applicable because of the big computational limit constraint.

Furthermore, the hardware of these kind of vehicles is usually an embedded system with small computation capacity compared to the current personal computers. Besides, they must work in real time, because of vehicle safety.

Of course, the algorithm has not to be a bottleneck to the hardware, because, it is a help tool for drivers, it is not an automatic drive system.

That means that the loop algorithm needs to have a minimum frame per second (fps) limit, in other words, each iteration should not exceed the maximum repeating time. Therefore, 25 fps or more should be enough to make the processing in real time. That is why almost all methods are discarded from the approach, for example, probability based algorithms, phase congruence methods or active contours.

3.1 General design

This section describes the pipeline of algorithm and goes deeper on the explanation of how each part of it works. As it is mentioned before, the whole process is grouped into 5 separate methods of the area of computer vision called segmentation, detection, tracking, mark recognition and measurements.

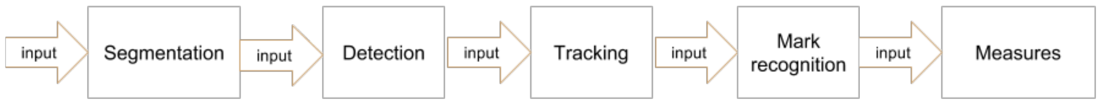


Figure 14: Computer vision sequence

In order to overcome the requisites of time and the software metrics and to obtain faster operations, it must optimize read-and-write operations in memory and reduce unnecessary code lines. They are divided the improvements into two global types: domain improvements and computation improvements.

Firstly, the domain models are usually based on algorithm constraints. For example, the bus moves forward and therefore, tracking movements will be only in one direction. Another example could be the metric size of white road marks. In this case, we know target size measures and it is possible to calculate the values of the pixels of the lane markings providing the connection between pixels and metric values.

On the other hand, due to the bus hardware constraints, the calculation improvements has been very helpful to make better loop algorithm. For example, thanks to high contrast of road between the image grayscale intensities and the simple geometries of lane mark, it brings a way to reduce the image resolution which in this case the size is $320x240$ pixels. By the way, the highest computation cost operations are the actions that require iterating many times through matrix. To overcome this problem for instance, in spite of taking the whole image, it only gets the half bottom of the frame which is the region of interest (ROI).

Of course, improvements lose the output information, but, the balance between computation and accuracy will be closer to real time the algorithm.

Along with this section, it will be described each task of process in depth and in the same order than is shown of figure 14.

3.2 Image Segmentation

The segmentation target discards every information which is not relevant for the application. Besides, it emphasizes the interesting information in a binarized image. This kind of binarized output images, in our case with white values for the critical elements and black values for background information. However, in many times, such as in other CV algorithms, in addition to removing the noise and show relevant information, some part of helpful information could lose, for example, in occluded parts or shadows. In order to get maximum speed in the algorithm, in addition to taking into possible account restrictions, it is necessary to define a segmentation method

In order to get maximum speed for the algorithm, in addition to taking into possible account restrictions, it is necessary to define a segmentation method which chooses the correct lane marking elements. So, this action will be designed for the road environment, and it will be taking to account these constraints:

- The lane mark positions should be placed in large contrast regions, it means that the step edge or intensity difference between a small group of pixels will be high.

- The lane marks have high-intensity values because of its white colour, so they are higher than the background.
- The coach camera is usually looking to the road so it is not necessary to apply the filter to the whole image.

The segmentation algorithm is called **BrightStepRowFilter** which it tries to find step edges for each row, and it paints in white colour the places where step edges are detected. The algorithm controls the derivative in *x-axis* direction to find these peaks. So, it has two input parameters defined by the user which they are called Δx for $x - axis$ width of derivative width and the maximum *threshold* value. The equation below shows a basic mathematical design of algorithm.

$$r = \tfrac{\Delta I}{\Delta x} \; c \tag{6}$$

where ΔI is the difference of intensities between the current point and $x - distance$ point.

The first input parameter value Δx depends on the image resolution, and therefore the control of this input parameters changes the output solution. If the value is high, there will be false positives because in the middle of this distance could be another step edge. On the other hand, if the value is low there could not find any step side because of the real images doesn't have high differences in terms of neighboring few pixels. The second parameter *threshold*, as it is shown before, it is static, and it controls the output r. Then, if the r parameter is higher than the threshold, the boundary will be selected as lane marking edge and the output function will be this.

$$f(n) = \begin{cases} 0 & \text{if } r < threshold \\ 255 & \text{if } r > threshold \end{cases}$$

The lane marks usually have two step edges, the first in the up direction and the next in down direction. In the middle of this process, the intensities should not vary too much because these beacons have fixed size, and it is possible to define minimum lane marking width size and maximum road marking size. This second process allows removing noise to the output. The algorithm 1 shows our segmentation method.

```
Data: BrightStepRowFIlter
Result: BinarizedImage BImage
initialization;
rowCounter=0;
colCounter=0;
tau=inputparameter;
th1=inputparameter;
th2=inputparameter;
while rowCounter¡BImage.rows do
    while colCounter¡dMatrix.cols do
        e2=ptRowSrc[colCounter];
        e3=ptRowSrc[colCounter+tau];
        dleft=double((e2-e1))/tau;
        found=false;
        if dleft ¿ th1 then
            found=true;
            while calculateVariance(dRight)¡th2 do
                BImage[colCounter][rowCounter]=255;
            end
            colCounter=getLastVariancePos;
        end
        rowCounter++;
    end
end
```

Algorithm 1: BrightStepRowFilter algorithm

To show the algorithm clearly, the figure 15 shows an example of intensity signal when the road mark is detected. It is divided in three parts, the first tries to detect a slope larger than the threshold, after it calculates the variance of intensities of each pixels, when the result value is over of the variance threshold, a slope down is detected.

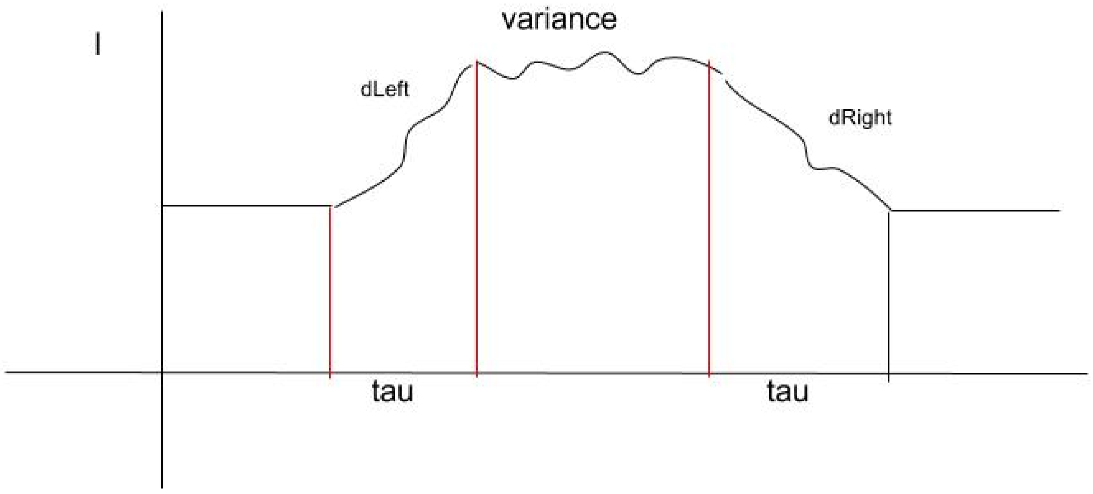


Figure 15: Image Signal figure

The processing output is a binary image where it means that the pixel values of the output image will be 255 white (our foreground) or 0 black (background). Finally, the output image could see in the picture below.

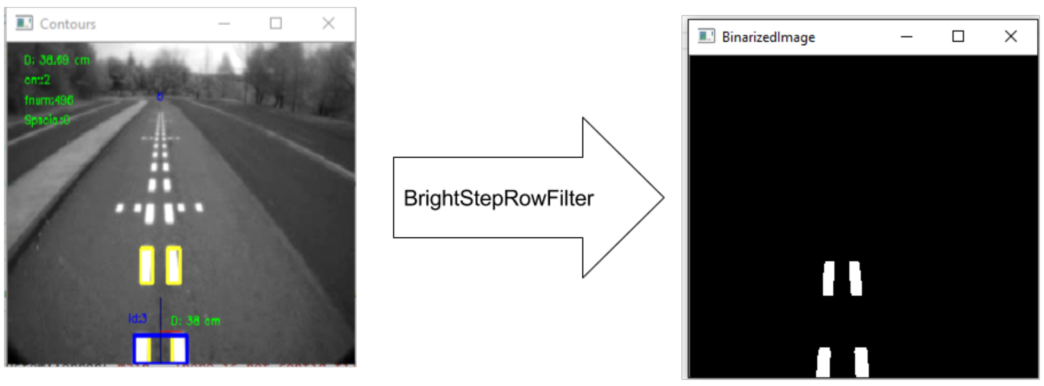


Figure 16: Segmentation output image

3.2.1 Contour detection

The segmentation process provides the opportunity to detect boundaries in matrices. The contour detector, that the developer uses, are based on OpenCV function *cv::findcontorus* whose paper is [31]. The approach of this article explains two border following algorithms proposed for the topological analysis of binary images. The first one is a extended version of border following algorithm which extracts topological information. Besides, it defines sorroundness relations between the outer edges and hole borders of a binary image. These extensions are:

- Border labelling puts an unique mark on each side in spite of taking on the same marking procedure from every border.
- And, it adds a method to obtain a parent contour of current following borders.

The second is a modified version of the first algorithm. It follows only outermost borders (the outer borders which are not connected with holes). The advantage is that it ignores the 1-component which are followed by other 1-component allowing to create a quick sequential method of 1-component counting. It is a good alternative from raster scan or labeling methods [18] [6].

The application executes the detection step for each input image, and gets an array of detected contours.

Each boundary transforms to contour polylines, and finally, these geometries convert into the bounding rectangles because tracking step doesn't work with polyline shapes.

Besides the detection area is the half down space of the image, in order to make an algorithm faster and to make easier the process to follow boundaries for each iteration.

3.2.2 Contour tracking

Although the detection step defines lane marking boundaries in x and y coordinates in time t, it is not able to make a connection between contour in time t and $t-1$ so, it does not recognize as a mark road region. This is exactly what the tracking step do. It calls with an identifier to any new contour which is appeared on the image and it recognizes its linear movement.

For each time t there is a contour detected that it stores a data structure (array) which contains cartesian coordinates, the start frame number and unique identifier. This array is compared with time $t-1$ array length creating nxm three matrices called distance matrix, mask and adjacent matrix such as (7).

$$\begin{pmatrix} t_{1,1} & t_{1,2} & \cdots & t_{1,n} \\ t_{2,1} & t_{2,2} & \cdots & t_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ t_{m,1} & t_{m,2} & \cdots & t_{m,n} \end{pmatrix} \tag{7}$$

Each matrix stores different information. The first is an euclidean distance matrix between objects in the previous and current tracking. The second matrix collects orientation and adjacency matrix between connections of objects.

These matrices work as the input and output parameters in the tracking algorithm which is divided into three main steps.

1. Calculate distance matrix.
2. Calculate Adjacency matrix.
3. Select the future objects of track constrained to:
    - Add new object to track if it was not in $t-1$ time.
    - Remove object of track if it was it $t-1$ time but it was not in t.
    - Update object position in the track.

These steps are converted to programming functions called $calculateDMatrix$, $calculateAdyacenceMatrix$ and $SelectFutureCandidates$.

The first takes as an input the current element of track. The output parameters instead are distance matrix and mask matrix. The input parameters of second function are these matrices (distance matrix and mask matrix) and the output parameter is the adjacency matrix which take this matrix and updates the candidates to current track.

The distance function calculates distances between point to point based on euclidean distance; that is, this function uses quadratic based function like (8) equation.

The Distance calculation function calculates distance between point to point based on euclidean distance; that is, this function uses quadratic based function like (8) equation.

$$D(n) = \sqrt{(d(t_x) - d(t-1_x))^2 + (d(t_y) - d(t-1_y))^2} \tag{8}$$

On the other hand, although the chosen distance calculation is euclidean distance. The mask matrix only stores result sign in the y axis direction, like the equations below.

$$M(t) = \begin{cases} d(t)_y - d(t-1)_y & \\ -1 & \text{if } M(t)_{x,y} < 0 \\ 1 & \text{if } M(t)_{x,y} > 0 \end{cases} \tag{9}$$

Thanks to this mask matrix which describes contour vertical movement direction in y axis, the algorithm has the knowledge to discard elements to adjacency matrix. The two matrices looks like the equations below.

$$\begin{pmatrix} 1 & -1 & -1 & -1 \\ 1 & 1 & 1 & -1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \tag{10}$$

$$\begin{pmatrix} 59.07 & 60.41 & 70.72 & 70.72 \\ 65.86 & 75.16 & 59.43 & 45.60 \\ 13.34 & 44.92 & 70.21 & 22.47 \\ 39.2 & 18.38 & 0 & 83.86 \\ 67.68 & 83.36 & 12.16 & 40.49 \end{pmatrix} \tag{11}$$

The second step consists of creating the association matrix between current tracking objects t and $t-1$ element. The mask contains 0 and 1 values. The zero value indicates that there is no connection between the selected row and column.

Each column refers to the current tracking objects detection list, and the rows describe foregoing tracked objects, and the result will store the relation between the past and present objects.

For example in the matrix in (12) whose size is 5×4, there are positive values on 3 locations, $M(3,1), M(4,2), M(5,3)$. So, each point, such as $M(3,1)$ means that the 3rd tracked element of last tracked frame is associated with the first tracking candidate object of the current frame.

However, there are entire rows or entire columns filled by 0-s. That means that there is no associated tracking element to the future candidates.

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{12}$$

For each row finds the minimum distance value and it selects as a future candidate. But this filter is not enough overcome the requisites because a $t-1$ row tracking object must not have more than one connections and vice versa.

But there could be a situation which the object, at t position of the track, has two or more one values. Therefore, the algorithm needs another filter which is capable of detecting more than one value per each column. Besides, it should select the cell of matrix which has only one value and minimum distance. The algorithm refalg:ady shows the fact described above.

```
Data: DistanceM and Mask
Result: AyacenceMatrix
initialization;
dmin=AdyMatrix[0][0];
indexX=-1;
indexY=-1;
while rowCounter¡dMatrix.rows do
    while colCounter¡dMatrix.cols do
        if dmin ¿ dMatrix[rowCounter][colCounter] and mask[rowCounter][colCounter]¡0
        then
            dmin=dMatrix[rowCounter][colCounter];
            indexX=rowCounter;
            indexY=colCounter;
        end
    end
end
setAdyRowTo0(rowCounter);
if indexX!=-1 and indexY!=-1 then
    AdyMatrix[indexX][indexY]=1;
end
indexX=-1;
indexY=-1;
dMin=dMatrix[rowCounter][colCounter];
while colCounter¡dMatrix.cols do
    while rowCounter¡dMatrix.rows do
        if AdyMatrix[rowCounter][colCounter]==1 then
            if dmin¿dMatrix[rowCounter][colCounter]) and
            (mask[rowCounter][colCounter]¡0 then
                min=dMatrix[rowCounter][colCounter];
                indexX=rowCounter;
                indexY=colCounter;
            end
        end
    end
end
setAdyColTo0(rowCounter);
if indexX!=-1 and indexY!=-1 then
    AdyMatrix[indexX][indexY]=1;
end
```

Algorithm 2: function: CalculateAdyacenceMatrix

Now, the adjacency matrix has the information to process the candidate selection feature with these steps.

- For each row, it tries to find a candidate. If there is a candidate, it updates the new tracking coordinates.
- If there is no candidate rows, it means that this tracking object must disappear; and therefore, it is removed.

- In each column, if there is not any candidate it implies that there is a new tracking object, so it creates a new boundary in the lane marking list.

However, the algorithm described before needs another layer to track the group of shapes to distinguish special distance marks and lane centre marks.

3.2.3 Grouping tracks

This process groups neighbouring tracking objects depending on these input parameters.

- The overall distance of nearby objects depends on the number of detections.
- The minimum distance between tracking objects.
- The variance on tracking at y axis and at upper left position.

For each frame, it only detects a unique group of tracks. For example, if the size of detections in a group is two, the distance is lower than the maximum length and the size of the track objects are similar, the tracked group can be a candidate to be a centre lane marking.

As long as the number of detections grows, it means that there are high possibilities to find a distance mark, as it shown in the figure 17.

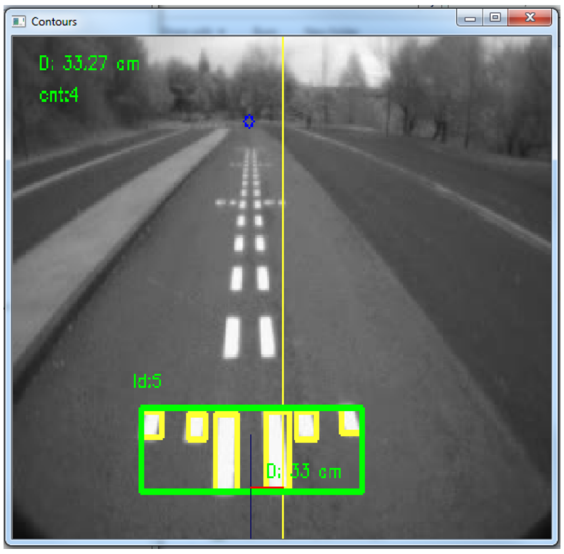


Figure 17: Group of distance mark at 15 meters

In addition to the position information, the group of tracks has more data such as creation frame number. If the group is active, the list of tracking objects increases with an ID identification. This information brings more knowledge to remove and count contours. Thanks to this information, it is an easy task to calculate longitudinal distance in meters, because each group size is at one meter. In conclusion, the sum of consecutive grouped contours will be the resulting distance in meters.

On the other hand, cross or transversal distance is more difficult. The next section explains in detail about transversal measure calculation method.

3.2.4 Taking Measures

This section explains how the algorithm takes the tracking information and calculates the ratio values which Datik requires.

The two output values of algorithm are:

1. Longitudinal distance of lane markings.
2. Transversal distance between image centre and lane, which is delimited by the road markings.

As described in above section, the first requirement is the easiest operation. It consists of the sum of elements in consecutive tracks. So, as long as new object is appeared and tracked, the distance counter amounts another meter to the global output.

The transversal distance calculation is more complicated because of this reasons:

- The center of image and the points at lane boundaries are defined in a projective space.
- The positions of tracked objects are defined as 2D image coordinates.

These two issues mean that it is necessary to establish a relation model between image coordinates and plane coordinates. In other words, the image coordinates must be converted into projective coordinates, because, in the projective space, we assume that every landmark point is in the same hyperplane which is called Homography. This transformation allows measuring the distance between the camera centre and the lane centre

The homography needs three points to create the road plane, the vanishing point, the image bottom right and bottom left. The vanishing point is located in the horizon, and we assume that every parallel lines in forward direction lie at this point. This coordinate changes over the time when the perspective view changes. As, for example, in the case of the curves. Due to this fact the vanishing point should be calculated in real time. However, in this project, the user defines the vanishing point statically, because the domain requires that every bus stop location is in a straight path.

Now the homography matrix has the world coordinate system, and it looks likes rotation and translation matrix $[R|\overrightarrow{t}]$.

$$Hm, n = \begin{pmatrix} R_{1,1} & R_{1,2} & R_{1,3} & t_{1,3} \\ R_{2,1} & R_{2,2} & R_{2,3} & t_{2,3} \\ R_{3,1} & R_{3,1} & R_{3,3} & t_{3,3} \end{pmatrix} \tag{13}$$

We assume that camera model is the same as the basic pinhole camera K.

$$Km, n = \begin{pmatrix} f & \gamma & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{14}$$

Where the f is focal length, u_0 and v_0 represent principal point (usually image center) and γ is a distortion coefficient (usually 0). So the result will be a projection matrix P.

$$P = K * [R|\overrightarrow{t}] \tag{15}$$

The monocamera computer vision methods have to face with a scale problem, and the homography needs this metric information to calculate the result with accuracy, and because this is necessary to define K intrinsics matrix for the calibration process.

There are different ways to do the calibration, it depends on what is the target of the application. The most popular method is called chessboard method. This method is common in the situations where there is no a priori knowledge of world information. As we knew, chessboard black and white rectangle sizes, it is an easy task to get camera and real world parameters, so, the homography stores the relation between the pixel-centimeter information.

Fortunately, as our space is a controlled environment the calibration process is more automatic than the chessboard task. The road where the bus stop process works always has the same lane mark pattern so, this a priori metric information knowledge can be compared with road markings pixel sizes. For example, if the dashed centre group track sizes are $(400, 500) = D_x, D_y$ mm and special distance marks are $(1400, 500) = D_x, D_y$, it allows to calculate the scale factor with pixels.

In conclusion, the steps to follows the algorithm are:

- Calibrate the default plane homography H with vanishing point and default predefined image points.

- Calculate the x scale $\frac{CurrentgroupTrackWidth}{D_x}$
- Calculate y scale factor $\frac{CurrentgroupTrackHeight}{D_y}$
- Update these new values to the homography.
- Finally the lane center distance from image center is calculated. Firstly the image center point Ic and the lane marking center point (Lc), are multiplied with the homography. Later it calculates the euclidean distance like the equation below.

$$\begin{cases} Ic = H(ic) \\ Il = H(lc) \\ HorizontalDistance = Ic_x - Il_x \end{cases} \tag{16}$$

3.2.5 Recognizing group of tracks

Although the primary objective of the project was to guide the vehicle in the stopping process, these figures bring another security layer to detect that the vehicle can recognize the bus stop when it is approaching the station. Besides it guarantees that the bus stop figures are correctly painted avoiding other confusing lane drawings. Mainly, it recognizes distance mark lateral geometric figures, such the following illustration.

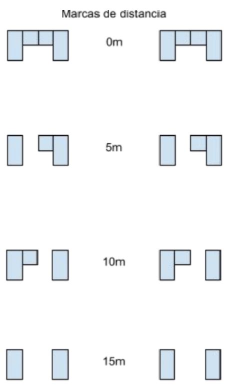


Figure 18: Distance geometric figures

As it can see in the picture below, the geometric figures are replicated in both sides. The first idea to overcome the classification issue was to use template matching technique, but unfortunately, it didn't work as well as expected. For example, the input images are in grayscale, and the binarized images do not bring enough feature points for a correct template matching. Therefore, there too much false positives in the recognition process. Besides the computation cost was huge for embedded devices.

Due to computation restrictions, the algorithm needs faster operations to work in real time, so the recognition process should be much more light and straightforward than template matching. Consequently, the author makes an algorithm based on summed histogram taking into account binarized image contrast information. Thanks to the algorithm have previously grouped the lane marks on bounding box; it reduces the search area of geometric road signs. Aforementioned arranged tracks have several parameters labelled, such as the position, area, unique identification, etc.

This task is only activated when a group of marks, more than two detections, appear into the scene. After, the algorithm processes the both sides replicated distinctive marks, and because of this, we will take into account the left mark to explain the algorithm, like the image below.

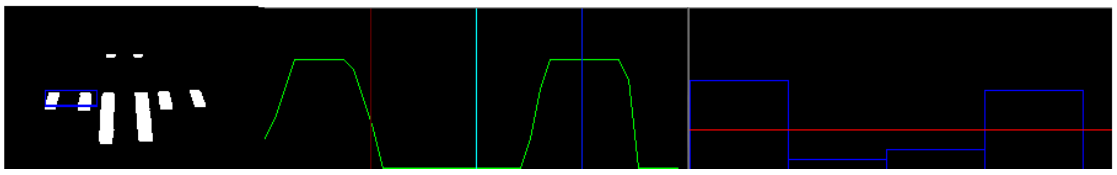


Figure 19: From left to right: binarized image, histogram sum graphic function, 4 bins divided mean histogram.

As it can see in the picture above, the binarized image distinctive mark (marked in blue) is grouped in two separated geometric figures. The following graphic function shows the pixel sum function evolution which, for each column, it displays the number of white pixel values in a green colour creating the full graphic along the matrix column axis. Besides the evolution graphic divides with four cells with the same size, which each cell represents different evaluation area. This division is targeted to evaluate the behavior in local space.

Then with this information, it creates a four bin new histogram and calculates the individual mean of each range. After that, this histogram bins compares the global mean of the function, helping to find the separating space between two geometric figures.

For example, in figure 21 the evolution function shows a global minimum value which represents the black space between two marks.

There are four types of lane marks, the first 15 meters road mark explained before, the 10*meter* and 5*meter* which they are similar, and final line mark.

The 5 and 10 marks are of the same type but they are inverted such as 20 and 21 pictures.

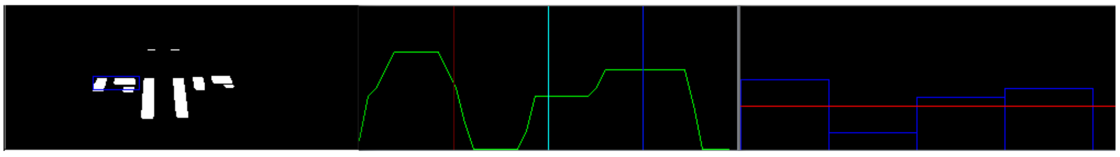


Figure 20: 5 meters mark

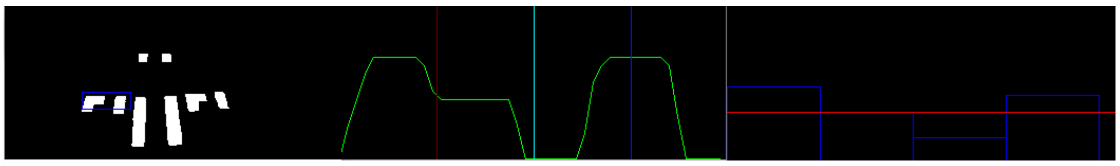


Figure 21: 10 meters mark

The mean of both graphics shows undoubtedly where is the black space, but besides the evolution function, for example in the picture 20, the first bin represents the global maximal and the second cell shows the decrease of white pixels. The third has the global minimal because it describes a hole or space, and finally, the last bin shows the local maximum value again.

In the same way, the final mark, fig 22, has two global maximum values in the first and the last cell, but it has minimum values in the second and third bins.

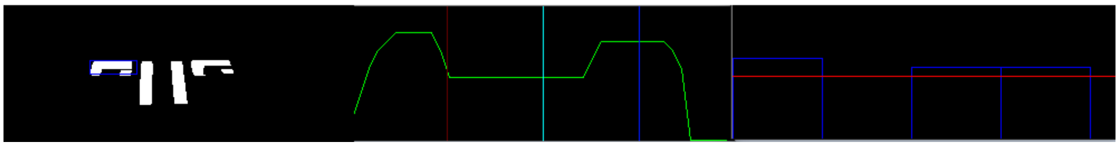


Figure 22: Final mark

Obviously, this algorithm it does not take into account the figures rotation, and it could be confusing if those shapes are vertically flipped because it will recognize flipped and not flipped pictures in the same way. This issue will have more cost computationally, and Datik agreed with this solution.

4 Experimental Results

The experimental results allow to evaluate the quality of our proposal algorithm in an elegant way. This section is separated into different parts with this order. Firstly, the setup describes a physical environment. The second part is about database description used for evaluation. Next, the author describes a evaluation strategy to understand the results values. And finally the objective results and subjective results are presented.

4.1 Setup

The setup step describes the experiments environment which are grouped on device features, database information and the physical setup.

4.1.1 Device features

The device features has two groups, hardware characteristics and software characteristics.

Hardware features:

- Quad arm cortex A9.
- 1 GB RAM.
- USB 2.0

Software characteristics:

- User defined and compiled Linux Yocto distribution for embedded systems.

4.1.2 Physical Setup

The app has been tested in real and virtual environments. Datik has no any real test scenario to test our algorithm. As a result, Vicomtech created a provisional bus stop place painting the corresponding markings taking into account the design documentation provided by Datik.

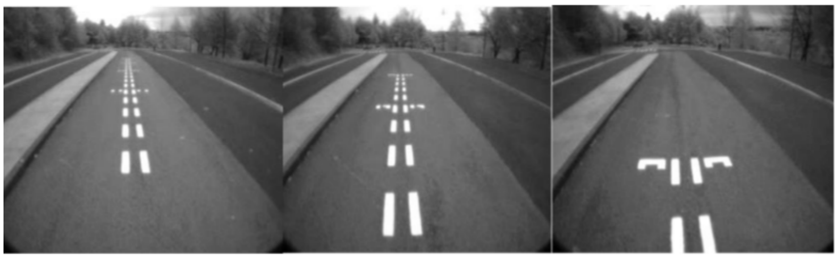


Figure 23: Real environment recordings

The videos were recorded in Miramon Parke Teknologikoa, in Donostia/San-Sebastian Gipuzkoa. Thanks to the test scenario is accessible from Vicomtech, the researcher made recordings in different use cases, simulating bus stopping motion types. The figure 24 shows the system prototype integrated in a car.

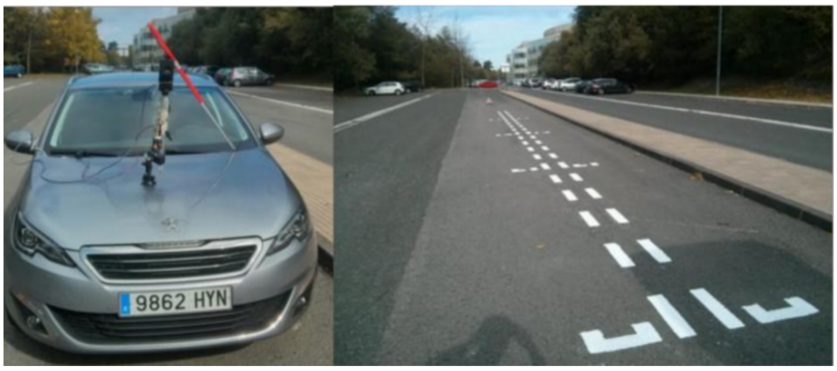


Figure 24: Physical car setup

On the other hand, the virtual simulations give the best scenario for start testing algorithm, because this world does not have the problems that contains the real world such as radial distortion, shadows or occlusions. The lens of virtual camera is 6 mm and it emulates the bus stop motion on the virtual road. The figure 25 shows a virtual environment example.

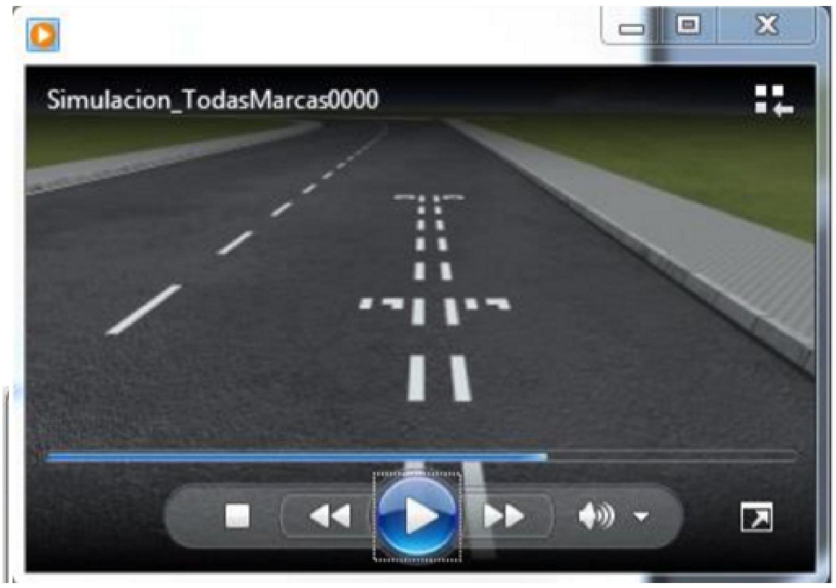


Figure 25: Virtual environment recordings

The table 2 defines the real and virtual world input parameters such as camera position height, distance and speed.

Table 2: Physical setup measures

Table 3: Real world

Height	distance	speed
1.48	car center	0-60 km/h

Table 4: Virtual world

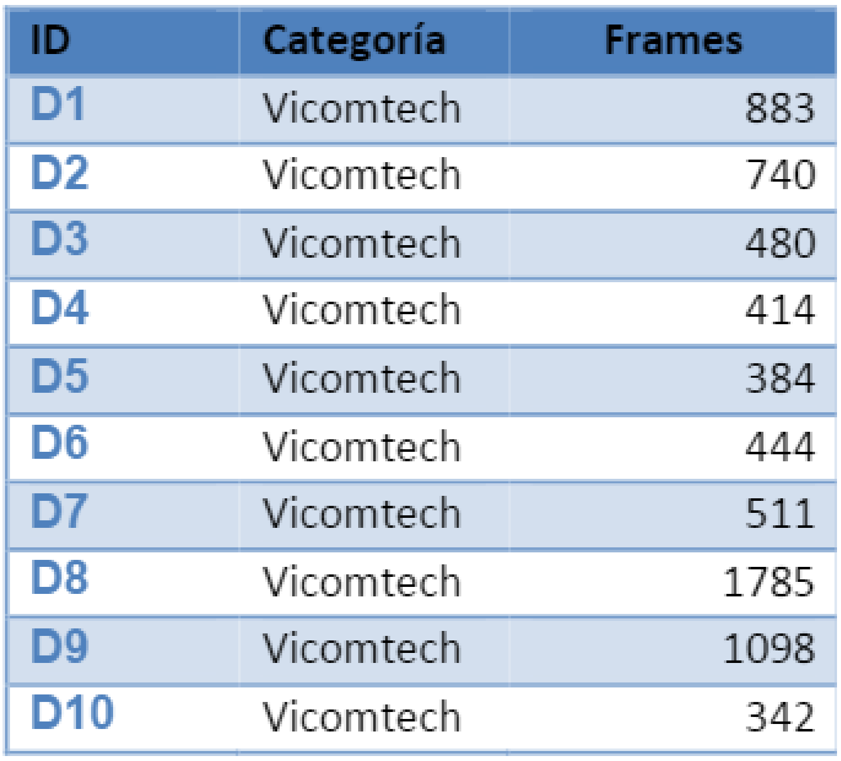
Height	distance	speed
1.50 m	image center	50 km/h

4.2 Dataset

The dataset is created by the real video recordings and virtual videos. But, even though, this data provides an input source to execute our algorithms, it is not possible to measure the error because it has no any labeled data to compare.

This labeling task is called annotation. These steps implies to describe functionality context, define a strategy and annotate frames.

The total duration of the dataset is 7621 frames, and the number of annotated videos are 10 as it can see in the figure 26. This corresponds to the recordings made in the Miramon Technology Park and the virtual world simulation.



ID	Categoría	Frames
D1	Vicomtech	883
D2	Vicomtech	740
D3	Vicomtech	480
D4	Vicomtech	414
D5	Vicomtech	384
D6	Vicomtech	444
D7	Vicomtech	511
D8	Vicomtech	1785
D9	Vicomtech	1098
D10	Vicomtech	342

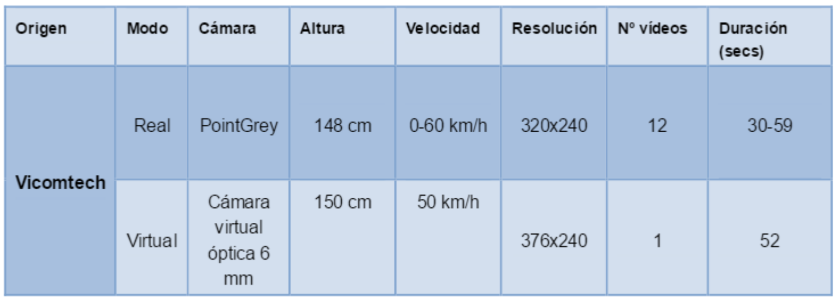
Figure 26: Annotation table: from left to right, Id, category, number of frames

4.2.1 Functionality context

This section defines the variables which contribute to an easier interpretation for the subjective results. The next list enumerates each parameter:

- Video Origin: Vicomtech.
- Modality: Real or virtual.
- Camera: Type of camera to make recordings in environments.
- Camera Height position.
- Image resolution.
- Driver bus speed.
- Number of videos.
- Video duration.

The data of the list above is reflected in the figure 27



Origen	Modo	Cámara	Altura	Velocidad	Resolución	Nº videos	Duración (secs)
Vicomtech	Real	PointGrey	148 cm	0-60 km/h	320x240	12	30-59
	Virtual	Cámara virtual óptica 6 mm	150 cm	50 km/h	376x240	1	52

Figure 27: Database general information

4.3 Evaluation Strategy

The evaluation process helps to the developer to show the quality of the application concerning precision, robustness and accuracy. That means that there is an annotated database to compare the algorithm output. This database is called ground truth GT and it usually consists on frame annotation.

There are annotated databases which are validated by the expert, but in this case, the road area marks are not universal. Therefore, it is necessary to create and annotate a database. When the annotation process finishes, in our case, it stores this information in *xml* file format. The algorithm output also saves the same *xml* file. Finally, it executes the evaluation algorithm matching GT and output files getting accuracy, preciseness and error results.

In the process of algorithm evaluation, the researcher has defined two strategies from different scenarios, one for special mark classification and the other, to measure the margin error between vehicle centre and lane centre.

The first one, it does not need any special annotation strategy because it is based on a binary classification of multiple classes used in pattern recognition. Therefore, it is possible to apply

Precision and Recall in the information retrieval scenarios. This multiclass classifier needs status annotation, which it means that it controls whether exists any geometric group for each frame. In other words, the user analyzes each frame trying to find a special mark. If he finds, he will annotate the status true in the corresponding category. Here, there is an example of an annotation.

```
<videoContentDescription>
    <video_data>
        <name>C:/proyectos/IECHB/Dataset/image_secuences/1_sequence/1</name>
        <frame_rate>25</frame_rate>
        <num_frames>883</num_frames>
        <width>320</width>
        <height>240</height>
        <flip>0</flip>
    </video_data>
    <object_list>
        <object id="10001" type="15Meters" frameStart="0" frameEnd="882" ">
            <status frameStart="496" frameEnd="504" keyFrame="1" value="1"/
        </object>
        <object id="10002" type="10Meters" frameStart="0" frameEnd="882" >
            <status frameStart="515" frameEnd="523" keyFrame="1" value="1"/>
        </object>
        <object id="10003" type="5Meters" frameStart="0" frameEnd="882" >
            <status frameStart="535" frameEnd="543" keyFrame="1" value="1"/>
        </object>
        <object id="10004" type="Final" frameStart="0" frameEnd="882">
            <status frameStart="557" frameEnd="566" keyFrame="1" value="1"/>
        </object>
    </object_list>
    <event_list/>
    <relation_list/>
</videoContentDescription>
```

So, there is an specific *status xml* tag, which describes for each class when it is visible the geometric group of tracks. The time is calculated with frame numbers *frameStart* and *frameEnd*.

The second one is more complicated because the output error is a float value which implies that the binary method does not work. Therefore, it is necessary to define a different strategy to get the lane centre point. As it is visible in the figure 28, there is no feature point to select the centre point on the road plane, making it very difficult to define the position with precision.

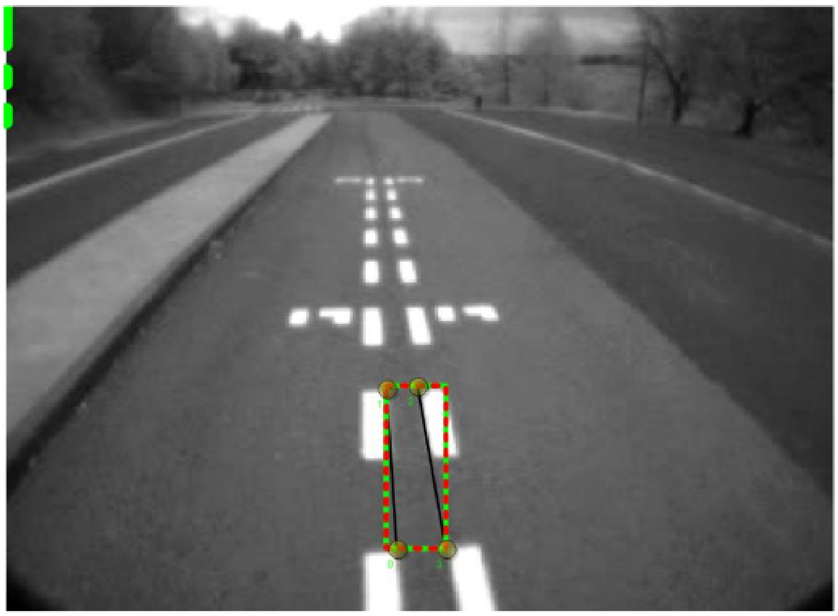


Figure 28: Sample polygon annotation

Fortunately, the centre lane marks give feature points and as a result it can calculate the centre point creating a polygon between the nearest marks, see figure 28.

The output will stores the polygon coordinates. The output file example is shown below.

```
<videoContentDescription>
<video_data>
<name>C:/proyectos/IECHB/Dataset/image_secuences/1_sequence/3</name>

<frame_rate>25</frame_rate>
<num_frames>480</num_frames>
<width>320</width>
<height>240</height>
<flip>0</flip>
</video_data>
<object_list>
<object id="10000" type="No-Class" frameStart="344" frameEnd="413"
annotate="Object n. 1" objectDataType="7">
<polygon frameStart="344" frameEnd="344" keyFrame="1">
    <point x="136" y="179"/>
    <point x="136" y="135"/>                <
    <point x="147" y="136"/>
    <point x="151" y="179"/>
</polygon>
</object>
</object_list>
```

After the annotations finish, the evaluation process starts. On the one hand it shows relevant information of classification. On the other hand, it evaluates the error between lane centre and bus centre.

4.4 Objective results

The objective results are divided into two parts. The first one, it calculates the distance error between the bus and the lane center, and the second one, it evaluates the classification error of special mark.

4.4.1 Distance error evaluation

This evaluation allows to analyze if the precision of the algorithm is good enough for our proposal requisites. For this kind of evaluations, output values are not based on binary evaluation which is relatively easy to define the success rate. In this case, we need to define metrical error parameters such as maximum error and mean error. The results can be seen in the table 5.

Table 5: Objective results

	Mean Error (cm)	Max error (cm)
D1	0.72	9.47
D2	1.75	90.15
D3	0.66	2.64
D4	5.10	96.97
D5	21.61	229.83
D6	1.40	19.17
D7	1.66	22.87
D8	1.59	53.16
D9	1.15	5.22
D10	0.51	3.74

Almost all datasets do not exceed 2 cm error except D5 and D4 where the error is the largest. The datasets D4, D5 and D6 makes sinusoidal movements and lane changes, which implies that this kind of manoeuvres directly affects to the outcomes.

The objective evaluation is a good start point to see the drifting error, but the result values do not describe its behavior. The section 4.5 helps to understand the evolution of the algorithm from another viewpoint.

4.4.2 Classification evaluation

The classification results allow analyzing the special lane marking detection success ratio.

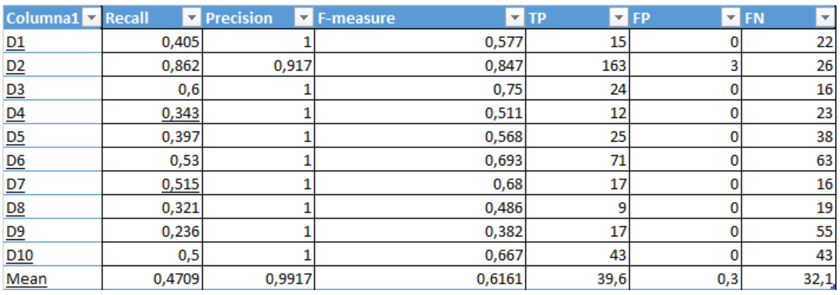
The section 3 explains how the researcher divides the different geometric marks into different classes and how the algorithm works to decide which figure corresponds to which categories.

The evaluation measures have three output values, *Precision*, *Recall* and *F-Measure*. These values work with some predictive variables, exactly True Positive *(TP)*, False positive *(FP)* and False Negative *(FN)*. The true positive is when the case is annotated and the system detects it. The false positive value, commonly called false alarm, is the case that the system detects an event, but it is not annotated (false event).

Finally, the false negative case is when there is an event annotated in the ground truth but the output algorithm fails in the detection.

The table 29 below shows the evaluation results, but firstly the Precision, Recall and F-measure values should be explained before.

- Precision: The fraction of positive instances which are relevant, usually called positive predictive value $P = \frac{TP}{TP+FP}$
- Recall: The true positive rate or sensitivity, in few words, the ratio between false negatives and true positives $R = \frac{TP}{TP+FN}$
- F-measure: It combines precision and recall with the harmonic mean $F = 2 * \frac{R*P}{R+P}$.

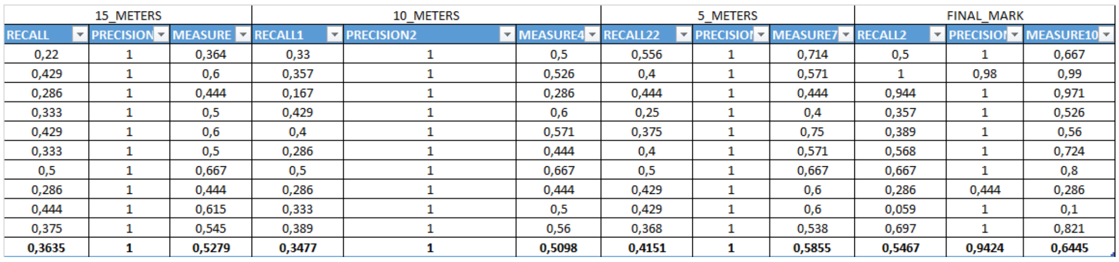


Columna1	Recall	Precision	F-measure	TP	FP	FN
D1	0,405	1	0,577	15	0	22
D2	0,862	0,917	0,847	163	3	26
D3	0,6	1	0,75	24	0	16
D4	0,343	1	0,511	12	0	23
D5	0,397	1	0,568	25	0	38
D6	0,53	1	0,693	71	0	63
D7	0,515	1	0,68	17	0	16
D8	0,321	1	0,486	9	0	19
D9	0,236	1	0,382	17	0	55
D10	0,5	1	0,667	43	0	43
Mean	0,4709	0,9917	0,6161	39,6	0,3	32,1

Figure 29: Global classification results

The figure 29 shows a relevant information. On the one hand, the recall is very low (too much FN) and the precision is very high. This means that there is a low false positive rate but the false negative rate is very high. In few words, the algorithm success rate when the processing task detects a special mark is very high, but it loses so much annotated special marks.

This global results give a first look of the classification task, but to see in more detail which class has the highest recognition rate, the classification results has been divided into 4 categories which each category is assigned to a special mark like the figure 30.



15_METERS			10_METERS			5_METERS			FINAL_MARK		
RECALL	PRECISION	MEASURE	RECALL1	PRECISION2	MEASURE4	RECALL22	PRECISIOI	MEASURE7	RECALL2	PRECISIOI	MEASURE10
0,22	1	0,364	0,33	1	0,5	0,556	1	0,714	0,5	1	0,667
0,429	1	0,6	0,357	1	0,526	0,4	1	0,571	1	0,98	0,99
0,286	1	0,444	0,167	1	0,286	0,444	1	0,444	0,944	1	0,971
0,333	1	0,5	0,429	1	0,6	0,25	1	0,4	0,357	1	0,526
0,429	1	0,6	0,4	1	0,571	0,375	1	0,75	0,389	1	0,56
0,333	1	0,5	0,286	1	0,444	0,4	1	0,571	0,568	1	0,724
0,5	1	0,667	0,5	1	0,667	0,5	1	0,667	0,667	1	0,8
0,286	1	0,444	0,286	1	0,444	0,429	1	0,6	0,286	0,444	0,286
0,444	1	0,615	0,333	1	0,5	0,429	1	0,6	0,059	1	0,1
0,375	1	0,545	0,389	1	0,56	0,368	1	0,538	0,697	1	0,821
0,3635	**1**	**0,5279**	**0,3477**	**1**	**0,5098**	**0,4151**	**1**	**0,5855**	**0,5467**	**0,9424**	**0,6445**

Figure 30: Classification results per class

The Precision value is very high on each class, so the results are the same of global analysis. But Recall values changes on each class. For example, the highest Recall values are in final mark and the worst results are in the 15 meters mark.

Obviously, the Recall values are not good, but this False Negative rate could come because the algorithm tracks the special patterns when it is very near to the bus, 4 or 5 frames before the mark disappears. The annotation, instead, starts when a special mark are in the half-down part of the image.

4.5 Subjective results

This section brings another viewpoint about algorithm behavior where it is not possible to explain with the result numeric values.

It is interesting to analyze the distance error graphics because it allows to see how the algorithm works. So, the graphic results will be represented in two figures for each use case with the following parameters.

- Comparison image between ground truth (Red) and algorithm output (blue).
- Quadratic error between ground truth and algorithm output.

There are 4 types of use cases which are the most common driving situations.

4.5.1 Case 1:

The first type of case is when the bus is driving in the straight direction and finally it stops in the final special mark. This is corresponded to **D1** dataset. According to figure 31, this is the case that it has the best performance and accuracy.

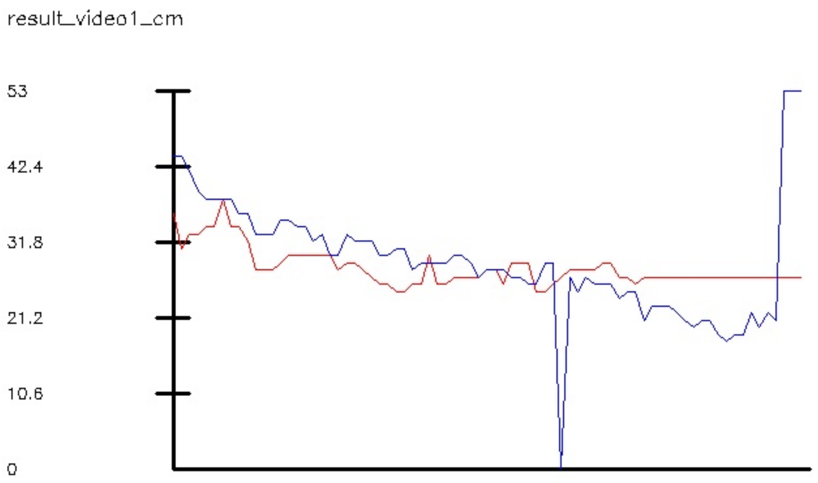


Figure 31: D1 dataset result graphic comparative

In the figure 31 there are visible noise peaks where it can be classified as outliers. This points directly impact on the calculation of the maximum error, as it can be seen in the figure below.

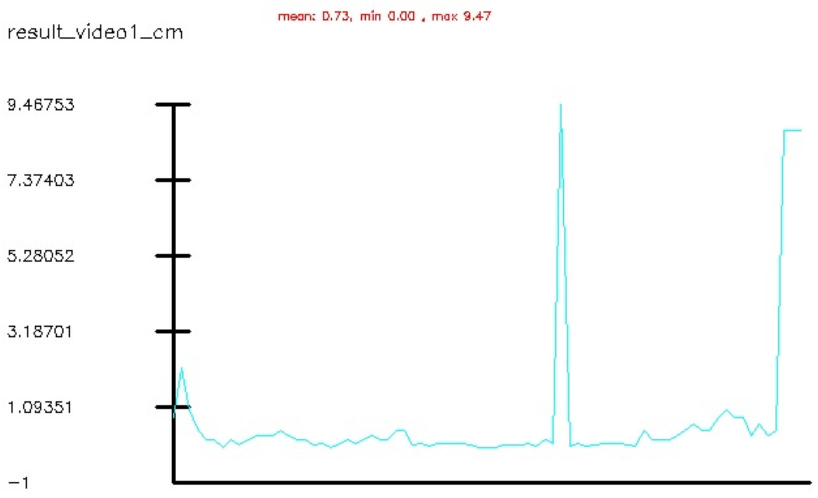


Figure 32: D1 dataset quadratic error

4.5.2 Case 2:

This is similar to the first case, but instead of drive in straight way it goes making zigzags and it corresponds to **D5**. Unfortunately, as it can see in the figure 33, this is the graph where it has the worst performance.

At the beginning of the graph, when the sinusoidal motion starts, the system begins to yield the largest errors, until it fits better results along the track. As in the previous case, there are peaks at particular moments of bad detections. The figure 34 shows clearly the noise peaks.

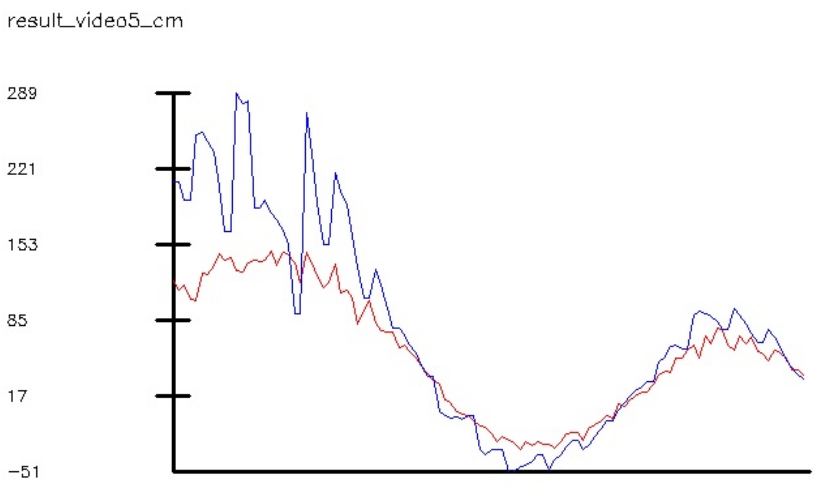


Figure 33: D2 dataset result graphic comparative

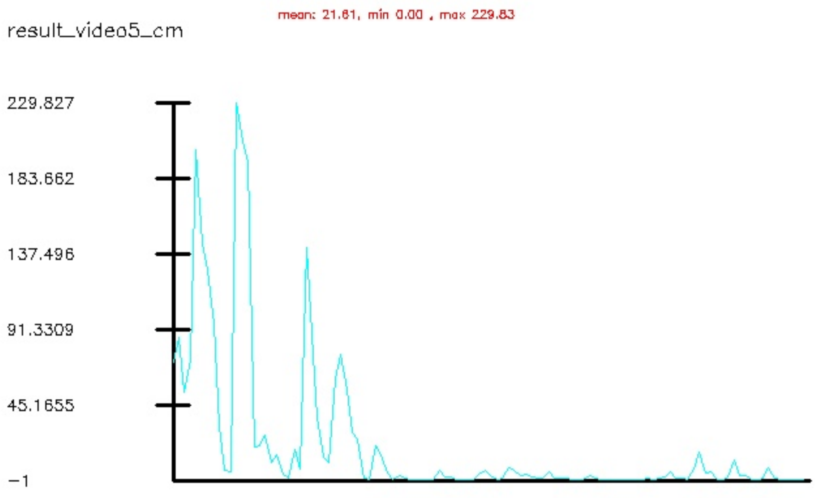


Figure 34: D2 quadratic error

4.5.3 Case 3:

The third situation has a lane changing to make bus stop and it drives in straight direction and the referenced dataset is **D6**. The results are displayed on figure 35.

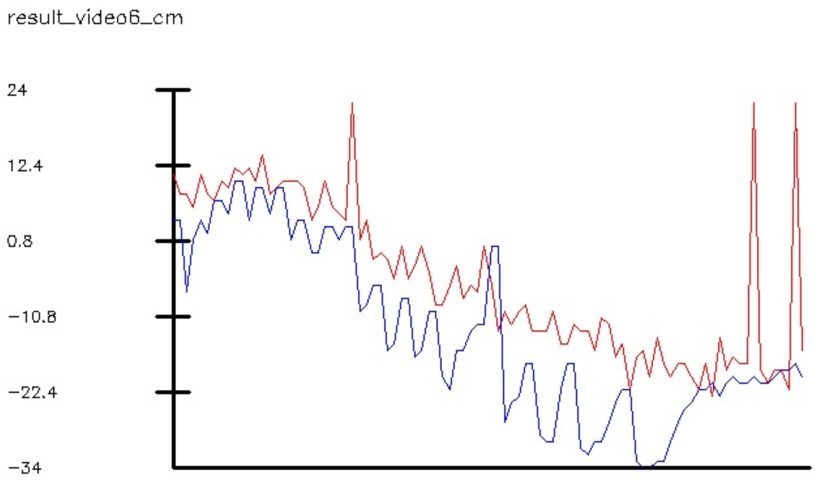


Figure 35: D3 result graphic comparative

Although it seems that functions are rather different, the difference in metric magnitudes is reduced compared to the case 2. There are peaks in the annotations because the camera sometimes loses frames. Although the annotation is correct, the calculation of the line centre marks causes the error. To view the graph of error of process, see figure 36.

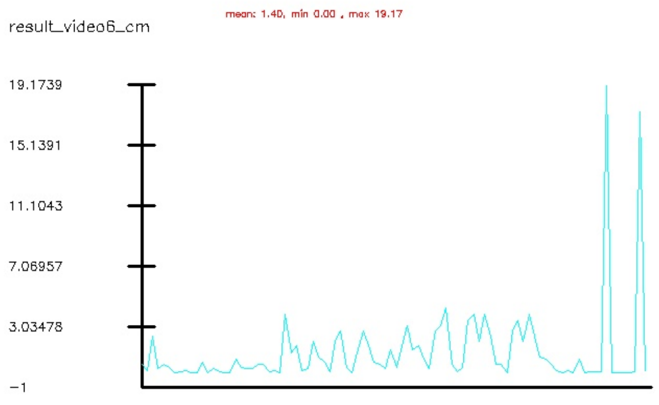


Figure 36: D2 quadratic error

4.5.4 Case 4:

Finally, the fourth case makes two rounds to the bus station without stop, and its dataset is called **D9**. The aim of this instance is to handle tracking and not tracking states. In figure 37 it is easy to notice where the tracking states start and end. When the algorithm is in *tracking state*, it stores the last read of the road that there is a horizontal line displayed in the middle of the graph.

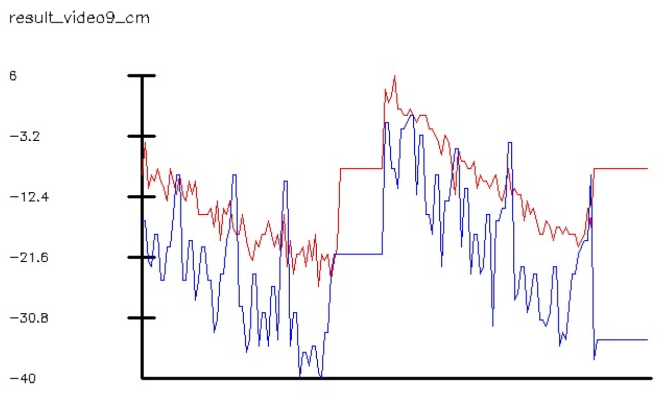


Figure 37: D4 result graphic comparative

The noise error is maintained constantly with few measurement error until the last part of function where there is a big outlier point because of the bad reading detection.

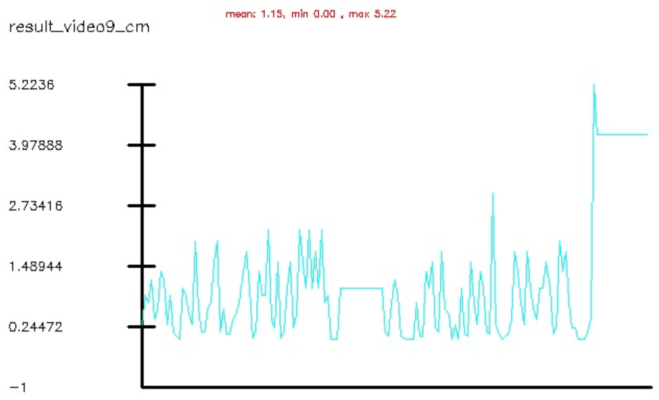


Figure 38: D4 quadratic error

5 Conclusions

This paper has presented computer vision algorithm to apply to Guided Bus Rapid Transit systems as an image processing module. Along the document, the researcher provides many mechanisms to overcome the computer vision troubles encountered in the different steps of the process: segmentation, detection, contour tracking, classification and distance calculation.

On the other hand, there does not exist any database with which to compare with our special lane markings. Therefore, Vicomtech has generated a database making video recordings.

Besides, Vicomtech staff painted the special lane markings on the road creating a test scene to make video recordings. Moreover, they also created a bus system physical structure emulating bus camera position, such as camera height and orientation angle.

The first version of the algorithm is able to overcome the required features such as, the longitudinal lane marking count, the transversal distance calculation and the special marking classification according to 3 and 4 sections.

In conclusion, the evaluation shows that the first version of the algorithm is precise enough because it respects Datik startup constraints (2cm +/-) and besides the algorithm is enough fast (77 frames per second) to execute in multiplatform embedded systems such as Windows embedded systems and Linux kernel based systems.

6 Future Work

This section explains the roadmap of the next steps in the future focusing on the next task. On the one hand, the target of the project is to improve the result precision in the classification task and the cross distance calculation. In the subjective results 4.5, it is evident that the noise peaks distorts the result values which it will be possible to solve in a straightforward way using Kalman filtering. This filter will allow getting smoother result function removing almost all noise.

About classification, in the section 3, it is mentioned that the special figure flipping is not taken into account, so there will be another way to overcome this problem in the future.

On the other hand, this algorithm is orientated to work in straight direction bus stops, which it means that it works with linear models and planes, but there might be some stations where the bus makes the stopping process in a curve. There could be possible to implement a parabolic model to recognize the curve lane, but it cannot affect the computation speed.

Finally, the vanishing point is a static input parameter. In the created database every video is recorded with the same calibrated system, but in case that the camera position and orientation changes, for example, when the bus typology is different that the simulation before, the calibration it must be done again. That is why a vanishing point autocalibration method will bring more portability to support any public transport vehicles.

References

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels. Technical report, 2010.

[2] Yasamin Alkhorshid, Kamelia Aryafar, Sven Bauer, and Gerd Wanielik. Road detection through supervised classification. *arXiv preprint arXiv:1605.03150*, 2016.

[3] JF Boyce, J Feng, and ER Haddow. Relaxation labelling and the entropy of neighbourhood information. *Pattern recognition letters*, 6(4):225–234, 1987.

[4] Mauricio Braga de Paula and Claudio Rosito Jung. Real-time detection and classification of road lane markings. In *Graphics, Patterns and Images (SIBGRAPI), 2013 26th SIBGRAPI-Conference on*, pages 83–90. IEEE, 2013.

[5] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[6] Roger LT Cederberg. Chain-link coding and segmentation for raster scan devices. *Computer Graphics and Image Processing*, 10(3):224–234, 1979.

[7] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):603–619, 2002.

[8] David Geronimo, Antonio M Lopez, Angel D Sappa, and Thorsten Graf. Survey of pedestrian detection for advanced driver assistance systems. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (7):1239–1258, 2009.

[9] Cosmin Grigorescu, Nicolai Petkov, and Michel A Westenberg. Contour and boundary detection improved by surround suppression of texture edges. *Image and Vision Computing*, 22(8):609–622, 2004.

[10] S-H Hsu and C-L Huang. Road sign detection and recognition using matching pursuit method. *Image and Vision Computing*, 19(3):119–129, 2001.

[11] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International journal of computer vision*, 1(4):321–331, 1988.

[12] Stan Z Li, Han Wang, and Maria Petrou. Relaxation labeling of markov random fields. In *Pattern Recognition, 1994. Vol. 1-Conference A: Computer Vision &amp; Image Processing., Proceedings of the 12th IAPR International Conference on*, volume 1, pages 488–492. IEEE, 1994.

[13] A Lopez, C Canero, J Serrat, J Saludes, F Lumbreras, and T Graf. Detection of lane markings based on ridgeness and ransac. In *Intelligent Transportation Systems, 2005. Proceedings. 2005 IEEE*, pages 254–259. IEEE, 2005.

[14] A López, J Serrat, C Cañero, F Lumbreras, and T Graf. Robust lane markings detection and road geometry computation. *International Journal of Automotive Technology*, 11(3):395–407, 2010.

[15] Shyjan Mahamud, Lance R. Williams, Karvel K. Thornber, and Kanglin Xu. Segmentation of multiple salient closed contours from real images. *IEEE transactions on pattern analysis and machine intelligence*, 25(4):433–444, 2003.

[16] David Marr. Vision freeman. *San Francisco*, page 95, 1982.

[17] Tim S Meese, Robert J Summers, David J Holmes, and Stuart A Wallis. Contextual modulation involves suppression and facilitation from the center and the surround. *Journal of Vision*, 7(4):7–7, 2007.

[18] David L Milgram. Constructing trees for region description. Technical report, DTIC Document, 1977.

[19] M Concetta Morrone and DC Burr. Feature detection in human vision: A phase-dependent energy model. *Proceedings of the Royal Society of London B: Biological Sciences*, 235(1280):221–245, 1988.

[20] M Concetta Morrone, John Ross, David C Burr, and Robyn Owens. Mach bands are phase dependent. *Nature*, 324(6094):250–253, 1986.

[21] Alan V Oppenheim and Jae S Lim. The importance of phase in signals. *Proceedings of the IEEE*, 69(5):529–541, 1981.

[22] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *Automatica*, 11(285-296):23–27, 1975.

[23] Giuseppe Papari, Nicolai Petkov, and Patrizio Campisi. Artistic edge and corner enhancing smoothing. *IEEE Transactions on Image Processing*, 16(10):2449–2462, 2007.

[24] Nicolai Petkov and Michel A Westenberg. Suppression of contour perception by band-limited noise and its relation to nonclassical receptive field inhibition. *Biological cybernetics*, 88(3):236–246, 2003.

[25] Wang Rongben, Guo Lie, Tong Bingliang, and Jin Lisheng. Monitoring mouth movement for driver fatigue or distraction with one camera. In *Intelligent Transportation Systems, 2004. Proceedings. The 7th International IEEE Conference on*, pages 314–319. IEEE, 2004.

[26] Christian Ronse. On idempotence and related requirements in edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(5):484–491, 1993.

[27] Azriel Rosenfeld, Robert A Hummel, and Steven W Zucker. Scene labeling by relaxation operations. *IEEE Transactions on Systems, Man, and Cybernetics*, (6):420–433, 1976.

[28] Hanno Scharr. *Optimal operators in digital image processing*. PhD thesis, 2000.

[29] Irwin Sobel and Gary Feldman. A 3x3 isotropic gradient operator for image processing. *a talk at the Stanford Artificial Project in*, pages 271–272, 1968.

[30] Joachim S Stahl and Song Wang. Globally optimal grouping for symmetric boundaries. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 1030–1037. IEEE, 2006.

[31] Satoshi Suzuki et al. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, 1985.