Real time image segmentation and labelling based on reconfigurable architectures

Andrés Fuster-Guilló[1], Juan Manuel García-Chamizo[1], Jorge Azorín-López[1], Francisco Flórez-Revuelta[1]

Abstract. A real time vision system of images perceived in real environments based on reconfigurable architecture is presented. The system provides surface labelling of the input images of unstructured scenes, irrespective of the environmental lighting or scale conditions adapting the response to temporal restrictions. The nucleus of the system is based on querying self-organizing maps constructed with supervised training by means of descriptors extracted from images of different surfaces perceived for successive values of the optical parameters (lighting and scale). To improve the labelling process the system estimates the environmental optical parameters and then self-organizing maps are reconfigured. A segmented architecture is proposed for the central module of the labelling process, which will improve the performance of image sequences. A prototype implemented on FPGAs applied as a guidance aid for vehicles is provided. This prototype can regulate its processing frequency at the speed demanded by the vehicle.

1 INTRODUCTION

In many application areas is necessary the development of a device with autonomous and powerful vision, capable of assisting mobile robots in a heterogeneous and changing environment. Autonomy implies a capacity to meet the requirements of the device without the need for a remote platform, fitting the system to the real time restrictions imposed by the problem and processing the images at the required frequencies . With regard to its power, is should provide operating capacity in realistic environments with changing or non-uniform light, where the scale of perception can change according to the distance, the different planes of the scene that remain out of focus, etc.

Within this framework, the overall objective of the research can be broken down into two partial aims: to provide a vision model of unstructured scenes in realistic conditions and an architecture that fulfils the performance expectations and the real time restrictions of the problem. The first one is related to the operating capacity in environments where visibility conditions (lighting, resolution, focus, ...) are variable and/or inadequate. The second one presents the need for the design of architectures specialized in vision from the proposed models. This is the objective that will be discussed in this paper.

Vision problems in real conditions (lighting, scale, vision angle, …) have been discussed in literature at a low level with the classic pre-processing methods [1] and at an intermediate level by means of the segmentation and labelling of the objects in a real scene [2] [3]. In this context, the objective of providing a vision model of unstructured scenes in realistic conditions has been more widely discussed in [4, 5]. The model and its application in scene labelling using techniques that query databases with non-specific descriptors, which can be used systematically in a wide range of real conditions (lighting, scale, non-uniform focusing, ...) are discussed in detail. To be precise, the brightness histogram was selected due to its generality and tolerance with geometrical transformations [6]. For the simplification of database queries, the use of context information is proposed [7].

The objective that is discussed in greater depth in this article is the feasibility of the system from the point of view of performance and its flexibility according real time restrictions. In this respect, the wide consolidation of the self-organizing model [8] and the implementation possibilities at a low level have been the criteria in its choice as classifier. Among the digital implementations of the SOM model we can find two fundamental trends, systolic arrays and SIMD [9, 10, 11] which have inspired the proposal for the segmented design. The capacity for rapid prototyping of reconfigurable computation has been used [12, 13], whilst being aware that system performance will be improvable with an ASIC design. The prototype has been tested as an outdoor guidance aid for vehicles. This aid is based on surface labelling of the sequences of the incoming images, while the use of this labelling in specific applications (guiding of vehicles, robots, aids for the visually handicapped …) has been left open. The prototype architecture can be reconfigured fitting the real time restrictions fixed by the vehicle speed [14].

2 VISION SYSTEM ARCHITECTURE

The aim to focus this article on the design of specific vision architectures has already been mentioned. In this section, a summary is given of the model for vision in realistic conditions, with details and results of its application in [4] [5].

2.1 Problem Formulation

The problem of realistic vision is related to the influence of variables that represent the calibration of the vision device on image formation. Consequently, the problem can be formulated in terms of the influence of certain calibration conditions on the sensitivity of the camera. A given vision device, with a given calibration and in environmental conditions, has a sensitivity around the value of the variable on which it operates. That value is the so-called calibration point. The function that describes the behaviour of the device acquires values at an interval around the calibration point. Generally speaking, we can assume that the calibration function is different for another calibration point. There will be a calibration point that generates an optimum calibration chart for each vision device.

Let Ψ be the value of the input to a vision system (characteristic magnitude of the scene that is being interpreted), and let $\Lambda^{\alpha}=\Lambda({}^{\alpha}z_j)$ be the function that represents the calibration to the values ${}^{\alpha}z_j$ of the n variables that characterize the vision device (environmental conditions or characteristics of the vision device).
Then the vision device output could be expressed as a function of the input Ψ and the calibration of the device Λ^{α}.

$$ {}^{\alpha,\Psi}f = f\left(\Psi, \Lambda\left({}^{\alpha}z_j\right)\right) \quad \forall\, {}^{\inf}z_j \le z_j \le {}^{\sup}z_j \quad j = 1\cdots n \qquad \textbf{(1)} $$

For the same input Ψ and another calibration Λ^{β}, the vision device output will be:

[1] Dept. of Computing Technology, University of Alicante, Spain, email: fuster@dtic.ua.es

$$^{\beta,\Psi}f = f\left(\Psi, \Lambda^{\beta}\right) \quad (2)$$

The input Ψ (for example the surface distribution in the scene or the interpretation at object level) and calibration variables $\Lambda(^{\alpha}z_j)$ (for example illumination or scale levels, …) in vision systems have spatial distribution in the scene and, consequently, in the projected image. Therefore, we will be able to express $\Psi(x,y)$ and $z_j(x,y)$ as functions of the coordinates of the image.
In order to deal with real acquired scenes, we propose the transformations expressed in (3) (4) y (5). The use of these transformations at the intermediate level of vision systems improves the interpretation processes at high levels.

The interpretation of an input T^{Ψ}. This transformation model provides the region labelling function at surface level of the image $^{\beta,\Psi}f$, acquired with different values of the calibration function Λ^{β}.

$$\Psi = T^{\Psi}\left(^{\beta,\Psi}f\right) \quad (3)$$

The estimation of the calibration function T^{Λ}. This transformation model obtains the calibration function value Λ^{β} that generates the image $^{\beta,\Psi}f$ for a given input Ψ.

$$\Lambda^{\beta} = T^{\Lambda}\left(^{\beta,\Psi}f\right) \quad (4)$$

The synthesis of an image. The output $^{\beta,\Psi}f$ for another calibration could be synthesized from an input Ψ and the system output $^{\alpha,\Psi}f$ for one of the calibrations.

$$^{\beta,\Psi}f = T^{s}\left(\Lambda^{\beta}, {}^{\alpha,\Psi}f\right) \quad (5)$$

The interest of this research is that it proposes a general method for carrying out these transformations, independently of the studied variables z_j of the calibration function $\Lambda(^{\alpha}z_j)$. This approach enables us to achieve our aim of proposing a general architecture for realistic vision. The calibration variables could reflect, for example, illumination conditions of the acquisition: the method will allow us to interpret the input Ψ, to estimate the calibration function Λ^{α}, or to synthesize a new image $^{\beta,\Psi}f$ with improved illumination levels Λ^{β}, from an image $^{\alpha,\Psi}f$ captured in deficient illumination $\Lambda^{\alpha}=\Lambda(^{\alpha}z_{illumination})$. The same can be said for other variables, such as scale, angle, etc.

The complexity of the problem of obtaining the transformation T, is related to the possibility of obtaining identical or very similar results at the output of the vision device $^{\alpha,\Psi}f$, as a consequence of different combinations from the input. For example, the image of two different surfaces can be very similar in different illumination conditions, even in identical but deficient illumination conditions (clear or dark).

$$f(\Psi^{1}, \Lambda^{\alpha}) = f(\Psi^{2}, \Lambda^{\beta}) \quad (6)$$

In this paper, we will focus on the use of the transformations expressed in (3) and (4) to deal with the segmentation and labelling of a scene from an image.

2.2 Solution proposal

The formulation of the problem in the previous section is open and different methods could be proposed for transformation functions T (3) (4) and (5). In this paper the proposed general model of transformation T uses knowledge bases to infer the region labelling function Ψ (3) or provide the calibration function values Λ^{α} (4). In any case, the inference from image $^{\beta,\Psi}f$, with the image $^{\alpha,\Psi}f$ for different calibration values Λ^{α} being known, is suggested. We will call these database $DB(^{\alpha,\Psi}f,\Lambda^{\alpha},\Psi)$. Consequently, we could formulate the expressions:

$$\Psi = T^{\Psi}_{DB}\left(^{\beta,\Psi}f, DB\left(^{\alpha,\Psi}f, \Lambda^{\alpha}, \Psi\right)\right) \quad (7)$$

$$\Lambda^{\beta} = T^{\Lambda}_{DB}\left(^{\beta,\Psi}f, DB\left(^{\alpha,\Psi}f, \Lambda^{\alpha}, \Psi\right)\right) \quad (8)$$

The complexity of the database queries in (7) and (8) expressed in (6), could be simplified by obtaining context information, we can then assume that the range of a variable in a region of the space is limited. Consequently, if we can estimate the average value of the variable in any one region, the probability of exceeding the expressed range of the variable in the region is low. In other words, the probability of obtaining values out of the context is low. The database queries in (7) and (8) can be simplified by obtaining context information, estimating the values Λ^{α} or Ψ and querying the partial view of the databases for the known values of Λ^{α} or Ψ. We will call these partial views $DB_{\Lambda\alpha}(^{\alpha,\Psi}f,\Lambda^{\alpha},\Psi)$ or $DB_{\Psi}(^{\alpha,\Psi}f,\Lambda^{\alpha},\Psi)$.

For the input of the system Ψ, heuristics can be designed to detect the simultaneous presence of certain groups of materials in the scene (contexts). Given the heuristic nature of this problem, this study is focused on obtaining context information for the calibration variables. If we can affirm that the range of the calibration variables, which we expressed as $^{inf}z_j<^{\alpha}z_j<^{sup}z_j$ (1), in a certain region of the image ($x_a<x<x_b$; $y_c<y<y_d$), is limited to $^{i}z_j<^{\alpha}z_j<^{s}z_j$, with $^{inf}z_j<^{i}z_j<^{\alpha}z_j<^{s}z_j<^{sup}z_j$, we can affirm that in this region we only need to query the part of the database corresponding to this range $DB_{\Lambda\alpha}(^{\alpha,\Psi}f,\Lambda^{\alpha},\Psi)$ with $\Lambda^{\alpha}(^{i}z_1<^{\alpha}z_1<{}^{s}z_1,\ldots,{}^{i}z_n<^{\alpha}z_n<{}^{s}z_n)$.

In this proposal, the calibration function Λ^{α} will be previously estimated by means of (8). That will enable us to obtain the region labelling function Ψ more precisely by querying the partial view of the database $DB_{\Lambda\alpha}(^{\alpha,\Psi}f,\Lambda^{\alpha},\Psi)$ (9).

$$\Psi = T^{\Psi}_{DB,\Lambda}\left(^{\beta,\Psi}f, \Lambda^{\beta}, DB_{\Lambda^{\alpha}}\left(^{\alpha,\Psi}f, \Lambda^{\alpha}, \Psi\right)\right) \quad (9)$$

1. Pre-processing. Context information. Calibration estimation (8)

1.1. Initial surface labelling Scan the unknown image with a window W and classify it according to the best match found on the complete DB. Label W with the surface.

1.2. Calibration estimation Scan the unknown image using the window W as in step 1.1 and classify it according to the best match found in the partial view of the DB for the known surface. Label W with the calibration. Use a statistical parameter to find the mean of the calibration of one region of the image or of the complete image.

2. Processing. Final surface labelling (9)

2.1. Scan the unknown image with the same window W as in step 1 and classify each window according to the best match found in the partial view of the DB for the known calibration. Label the window W with the surface of the DB element.

Figure 1. Model phases for surface labelling

2.3 Architecture based on SOM

In order to tackle the classification task of the scan windows of the unknown image by comparing them with different images stored in databases, self-organizing maps have been used because of their discriminating capacity and high degree of parallelism inherent to connectionist methodologies. These self-organizing maps enable the discriminating capacity of different features extracted from the images to be evaluated, that is, their suitability for grouping the unknown images together in accordance with different classification criteria, such as region labelling or the calibration value. The architecture will be general and will enable the problems of realism introduced by the different calibration variables to be dealt with by means of reconfigurations of the self-organizing maps neurons. Although, self-organizing maps are only one of the options for the model, in this study we use them with supervised training to illustrate the implementation of the databases.

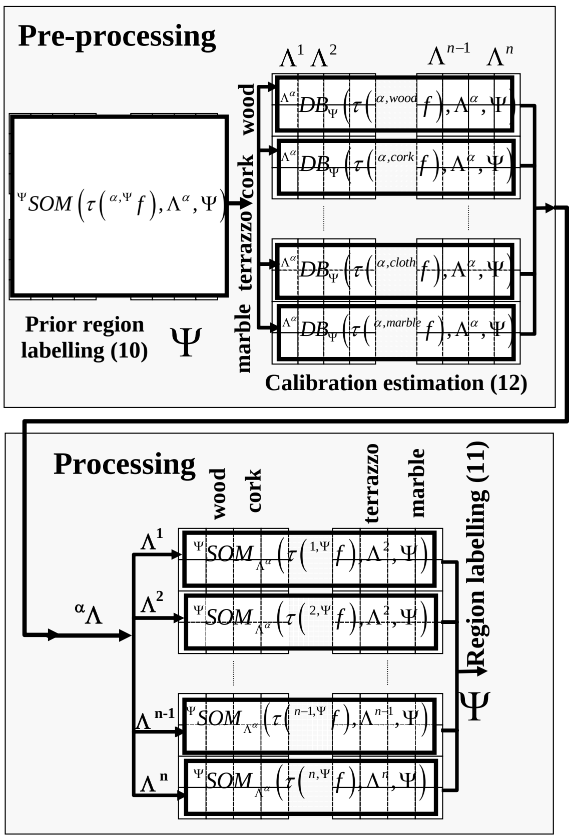

Pre-processing
Λ^1 Λ^2 Λ^{n-1} Λ^n
wood
cork
terrazzo
marble
$^{\Psi}SOM\left(\tau\left(^{\alpha,\Psi}f\right),\Lambda^{\alpha},\Psi\right)$
$DB_{\Psi}\left(\tau\left(^{\alpha,wood}f\right),\Lambda^{\alpha},\Psi\right)$
$DB_{\Psi}\left(\tau\left(^{\alpha,cork}f\right),\Lambda^{\alpha},\Psi\right)$
$DB_{\Psi}\left(\tau\left(^{\alpha,cloth}f\right),\Lambda^{\alpha},\Psi\right)$
$DB_{\Psi}\left(\tau\left(^{\alpha,marble}f\right),\Lambda^{\alpha},\Psi\right)$
Prior region labelling (10)
Ψ
Calibration estimation (12)
Processing
$^{\alpha}\Lambda$
$^{\Psi}SOM_{\Lambda^{\alpha}}\left(\tau\left(^{1,\Psi}f\right),\Lambda^{1},\Psi\right)$
$^{\Psi}SOM_{\Lambda^{\alpha}}\left(\tau\left(^{2,\Psi}f\right),\Lambda^{2},\Psi\right)$
$^{\Psi}SOM_{\Lambda^{\alpha}}\left(\tau\left(^{n-1,\Psi}f\right),\Lambda^{n-1},\Psi\right)$
$^{\Psi}SOM_{\Lambda^{\alpha}}\left(\tau\left(^{n,\Psi}f\right),\Lambda^{n},\Psi\right)$
Region labelling (11)

Figure 2. Whole architecture for image labelling in real conditions based on SOMs

The SOMs (10) (11) have been constructed from features extracted from the images $\tau(^{\alpha,\Psi}f)$ belonging to the database $DB(^{\alpha,\Psi}f,\Lambda^{\alpha},\Psi)$ (different materials Ψ with different calibration function values Λ^{α}) and classified according to material $^{\Psi}SOM(\tau(^{\alpha,\Psi}f),\Lambda^{\alpha},\Psi)$. The labelling of self-organizing maps per surface may provide success levels that indicate the suitability, in certain cases, of using the whole database to carry out region labelling Ψ (10).

$$\Psi = \mathrm{T}^{\Psi}_{SOM}\left(^{\beta,\Psi}f,\ ^{\Psi}SOM\left(\tau\left(^{\alpha,\Psi}f\right),\Lambda^{\alpha},\Psi\right)\right). \qquad \textbf{(10)}$$

Database queries can be simplified by prior estimation of the calibration values Λ^{α} and the subsequent query of the partial view of the databases $DB_{\Lambda\alpha}(^{\alpha,\Psi}f,\Lambda^{\alpha},\Psi)$. These database partial views will be classified per material $^{\Psi}SOM_{\Lambda\alpha}(\tau(^{\alpha,\Psi}f),\Lambda^{\alpha},\Psi)$. Once the calibration value Λ^{β} has been estimated, the map corresponding to this value is activated, as expressed in (11). These partial maps separated by calibration levels Λ^{β} avoid the overlapping of some patterns (6) and thus offer better results.

$$\Psi = \mathrm{T}^{\Psi}_{SOM,\Lambda}\left(^{\beta,\Psi}f,\Lambda^{\beta},\ ^{\Psi}SOM_{\Lambda^{\alpha}}\left(\tau\left(^{\alpha,\Psi}f\right),\Lambda^{\alpha},\Psi\right)\right). \qquad \textbf{(11)}$$

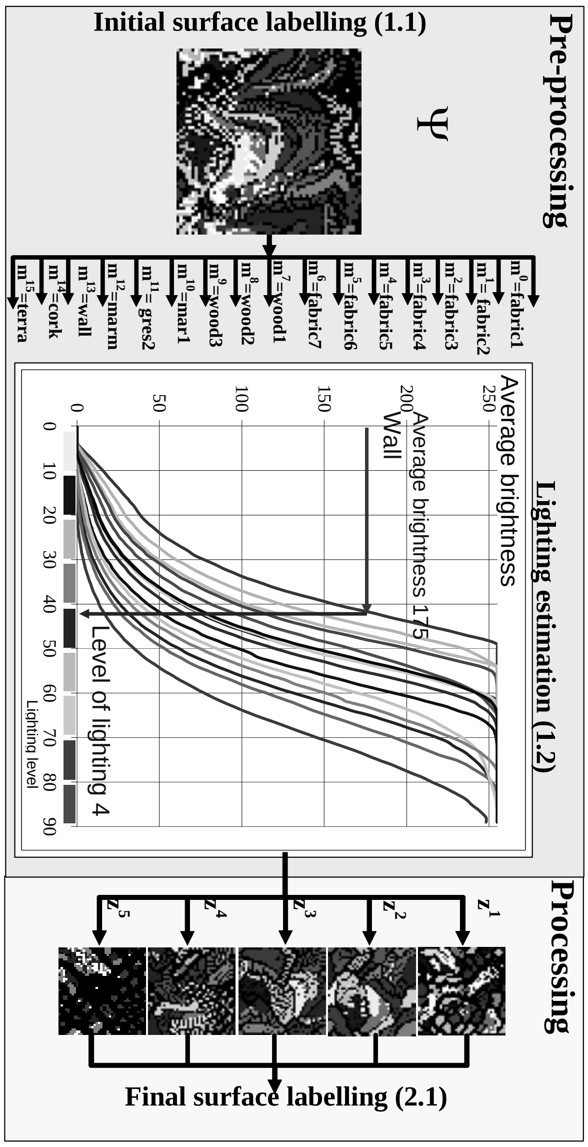


Figure 3. Architecture for surface labelling

In the pre-processing step, the calibration Λ^{β}, could also be estimated by means of database queries. Obtaining features of the images that allow the calibration to be estimated without knowing the superficial configuration, is a complex task. In addition, not all materials are as suitable for calibration estimation. We have previously established the interest in considering calibration to enhance region labelling. Now, we have observed the interest in knowing the region configuration to estimate calibration. In order to solve this paradox (Figure 1), first, the complete database is used to carry out prior region labelling (10). Next, the surfaces suitable for estimating Λ^{β} will be selected, and a database query of the partial view of the database for the known values of Ψ $DB_{\Psi}({}^{\alpha,\Psi}f,\Lambda^{\alpha},\Psi)$ is carried out (12). The use of SOMs in (12) depends on the complexity of the features for calibration estimation. In the last stage, once the calibration function value of a region is known as a statistical parameter of the elemental calibrations, the final enhanced region labelling is carried out by querying the partial views of the database for the calibration known (11).

$$\Lambda^{\beta} = \mathrm{T}^{\Lambda}_{DB,\psi}\left({}^{\beta,\Psi}f, \Psi, DB_{\Psi}\left(\tau\left({}^{\alpha,\Psi}f\right), \Lambda^{\alpha}, \Psi\right)\right). \qquad \mathbf{(12)}$$

3 REAL TIME SEGMENTED ARCHITECTURE OF THE VISION SYSTEM

In this section, the design of an architecture that fulfils the performance and provides real time image labelling capabilities is described in depth. The parallelization possibilities depend both on the descriptors used and on the implementation resources. In the studies carried out using the brightness histogram [4] [5], the tolerance of the model with regard to the scale and the need to process the lighting in the majority of cases have been observed. Consequently, pre-processing is confined to the estimate of the lighting level with the proposed interpolation schema. In addition, continual pre-processing is not necessary since the real values only change at specific time intervals. In this way, pre-processing (1) is only performed after a specific time interval has passed in order to bring about, should it be the case, the reconfiguration of the hardware necessary to the central part of the processing (2). We only give details of the processing architecture as it is the phase that is continually executed. However, the pre-processing architecture is very similar, with only the addition of the interpolation schema and voting per region. In order to deal with the problem of real time restrictions, we use a similar scheme obtaining the speed changes in preprocessing. Once this change has been detected, the architecture will be reconfigured with the necessary parameters.

Since the transformations perform on long window sequences of the images (the same number as scan windows and assuming the capture of successive images), a segmented architecture that performs different parts of the process on successive windows of the image is justified. The division of the process into phases should integrate the calculation of the histogram and the query of the self-organizing map. One possible segmentation of the histogram into phases assigns each phase with the processing of h of the m pixels of each window (with any dividing criterion for the window). With regard to the self-organizing map, each phase will calculate the minimum distance and the winning neuron of a subset s of neurons of the total r of the map.

One drawback to the segmentation of the histogram is the need to transfer the windows and the histograms from one phase to the following one. This requires storing them in intermediary registers, which results in a problem of space. In order to solve this problem, we have resorted to a SIMD schema for calculation of the histogram. Consequently, a hybrid architecture (Figure 4) has been proposed, with the histogram modules performing with the SIMD schema and the neuron modules processing in a segmented way. The adjustment of the number h of points corresponding to each histogram module and the number of s neurons corresponding to each neuron module will enable the duration of the histogram and neuron modules to be balanced in order to adapt it to the segmentation cycle. Also, parameters like the number of sampling windows or neurons of the SOM can be adapted to real time restrictions, although at the cost of losing quality in the labelling process.

The normalized brightness histogram used in the system is centered on average brightness, that is, the average brightness of the histogram of a window is centered on the brightness range (for example, centered on 128). This operation is called normalization. In addition, a component representing the average brightness is added to the histogram to compensate for the loss of information in normalization. In any case, this normalization process is complicated enough to constitute a different phase. The possibility of integrating the histogram calculation and the normalization increases the size of the clock cycle. Therefore, an architecture with two phases for the calculation of the histogram and subsequently a succession of phases that calculate the winning neuron for that window has been proposed. For implementation of the SOM, the Manhattan distance has been used to avoid multiplications and square roots of the Euclidean distance.

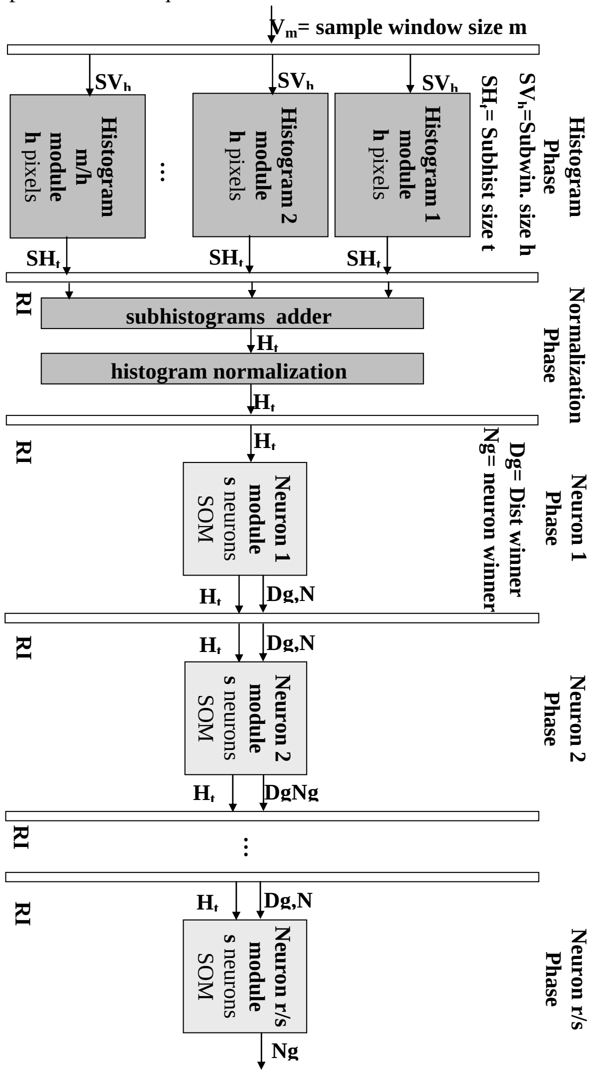


Figure 4. Real time segmented architecture for image labelling

4 PROTOTYPE IMPLEMENTATION

A rapid prototyping board has been used for implementation. To be exact, the development platform has been the RC1000 from

Celoxica. It is a card for PCs with PCI bus that consists mainly of a FPGA Xilinx Virtex XCV2000E and an 8 MB memory, arranged in 4 independent banks of 2 MB. The structure of the vision system can be seen in figure 5. It is made up of four functional units: *memory*, *host*, *uc*, *histogram* and *som*.

The *memory* stores the input image of the system and the image resulting from the labelling of regions in different banks. This allows us read and write the memory at the same clock cycle. The *host* is responsible for communication between the frame grabber and the FPGA and for the high level coordination of the system: exclusive access to memory, DMA transfers, etc. The *histogram* unit calculates the normalized histogram of the sample window. The *som* unit implements the queries of the self-organizing map. Finally, the *uc* generates the control signals necessary for the functioning and synchronization of the different units of the system. Two clock cycles are set up within the system: *clk* and *phase*. The *clk* clock comes from a FPGA pin connected to a programmable clock that runs in a frequency range of between 400 kHz and 100 MHz. The *phase* cycle is a frequency divider of the clock *clk*. With every *p* cycle of the clock *clk* we obtain a clock cycle of the *phase* segmentation.

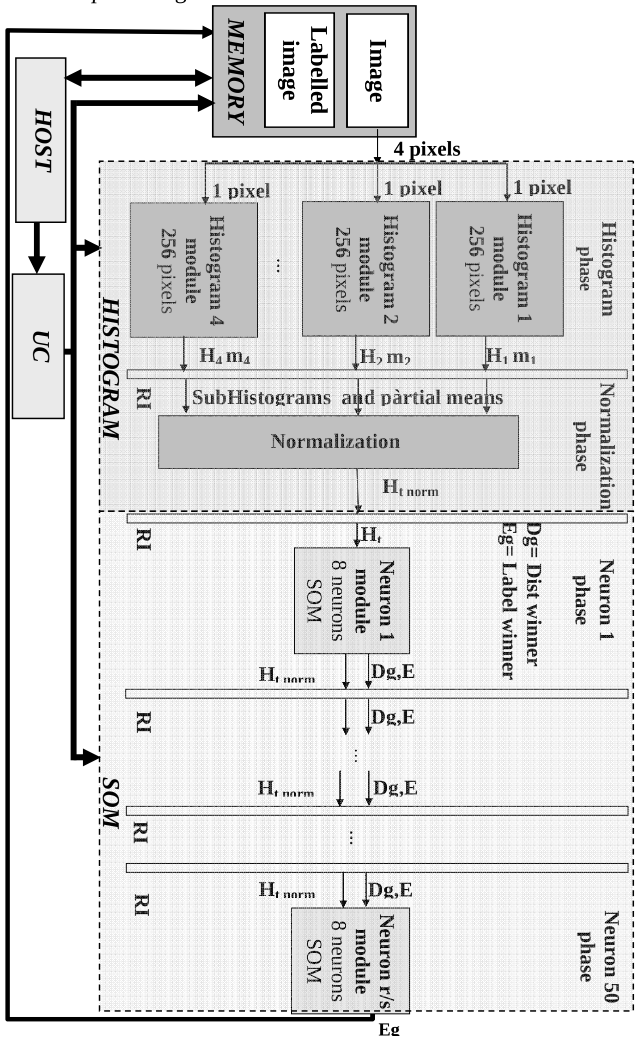


Figure 5. Prototype structure based on reconfigurable hardware

The functioning of the system begins with the capture of an image of 1024 x 1024 pixels of 8 bits (grey scale). This image is transferred via DMA to the memory of the system and control is ceded to the *uc*. In each segmentation step a window of the input image of 32 x 32 pixels (m = 1024) is calculated. The memory is able to provide a word of 32 bits for each access. Therefore, in each clock cycle (*clk*) the memory provides the *histogram* unit with four pixels of the sample window (see figure 5).

The *histogram* unit is made up of two segmentation phases: the *histogram phase* and the *normalization phase*. In the *histogram phase*, 4 modules are organized to calculate the accumulation of the frequencies of the histogram (subhistograms) in 32 (t) grey levels and the sum of the pixel values to obtain the mean for each subwindow. The division into four modules is determined by the reading restriction of a maximum of 4 pixels in a memory bank. Therefore, each of these modules will fill a subwindow of 256 pixels of the sample window (h = m / 4) and the duration of the *segmentation phase* will be set at 256 *clk* (p = 256). The *normalization phase* adds up the subhistograms calculated as well the partial sums of the mean, calculated by the modules of the previous segmentation phase. In addition, it shifts the histogram frequencies to the central component (16) and adds the mean to the last component of the histogram.

After two segmentation cycles have passed, the *som* unit has the normalized histogram of a sample window of the image in its input. This unit calculates the window label, which will be stored in the memory using *r/s neuron* modules. Each module is designed to query sequentially the winning neuron to a subset of *s* neurons of the SOM. The Manhattan distance between two vectors (the normalized histogram and the weights of the corresponding neuron) of 32 components (in 32 *clk* cycles) is calculated *s* times for each neuron, to give the minimum distance.

Therefore, the need to balance the duration of the histogram and neuron modules, to adapt it to the segmentation phase and given that the card architecture sets the *h* value at 256, the number of *s* neurons corresponding to each segmentation phase of the *som* is equal to 8. In equation 13, we can see the expression that relates the segmentation phase to the duration of the histogram, neuron and *clk* clock modules.

$$phase = h \cdot clk = s \cdot t \cdot clk \quad . \tag{13}$$

The segmentation phases necessary to calculate a sample window will depend on the size of the map. In this implementation, a 400-neuron map (r = 400) has been used, so after 50 segmentation phases the label of the window that is stored in the memory is obtained in the *som* module output. Finally, the labelled image is transferred from the *memory* to the *host*. The size of this image will depend on the sampling performed.

Let e be the number of segmentation phases, v the number of windows in the input image, *wi* the width of the input image, *wv* the width of the sample window and *fm* the sampling of the image (distance in pixels between one sample window and the next), then generally speaking, the time necessary to calculate an image in this system will be the one expressed in equation 2.

$$t = e + (v-1)\cdot phase \big/ \quad v = \left(\frac{wi - wv}{fm} + 1\right)^2 \quad . \tag{14}$$

The architecture parameters (e, v, phase) can be adapted looking for more efficient results and agreed the temporary restrictions that prevail. The readjustment of the number of neurons of the complete map can allow to reduce the number of stages and to obtain an earlier first answer; nevertheless the common situation is the presence of a large number of sample windows in the input, being

this response time of the first window less important. The phase it also can be reduced by means of the decrease of the number of neurons s in each stage of module SOM, as well as of the number of pixels h calculated in each module subhistogram. All these parameters could be readjusted to improve performance.
Nevertheless, it is the number of windows used in sampling v, the parameter that offers more possibilities for fitting it before temporary restrictions. This parameter can be reduced by increasing fm and allowing us to fit the labelling frequencies (labelled images/s) to the speeds of the vehicle (meters/s). Although, this speed control is obtained at the cost of providing less precise labellings.

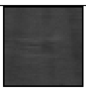
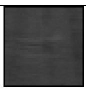
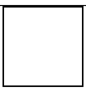
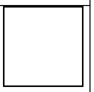
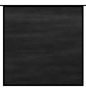
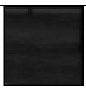
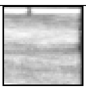
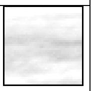
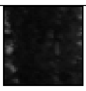
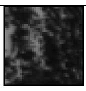
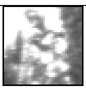
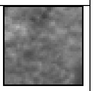
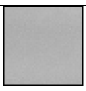
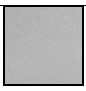
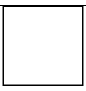
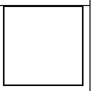
5 SYSTEM TESTING

Tests on the system as a guidance aid have been carried out on the pathways of the University of Alicante, where a moving vehicle equipped with a PC with a RC1000 card, capturer and camera (Figure 6) has been driven. The sequences of images processed by the system are captured using the camera situated at the front of the vehicle. System tests related to the accuracy rate of the surface labelling as well as the frequencies of image processing will be discussed in detail in the following sections.

5.1 Labelling Tests

The labelling success rates of the system are calculated by comparing the results with supervised labelling. The labelling tests of the static system is discussed in detail [4] and [5] for the scale and lighting processing, where an improvement in labelling as a result of the preprocessing can be seen, specially in the case of lighting. In the previously mentioned tests, a software simulation different from the hardware prototype was used. The differences in this prototype are the following: use of Manhattan distance, reduction in the number of histogram components to 32 and reduction in the number of map neurons to 400. The same tests have been repeated with the hardware prototype, with the accuracy rate being reduced from 92% to 90% in the case of preprocessing. In figure 6, the vehicle from which the image sequences were captured, some of the frames and the result of the surface labelling can be seen.

Table 1. Samples with different lighting conditions, extracted from the database used for the prototype of image labelling in outdoor scenarios.

	Lig.0	Lig.0	Lig. 1	Lig.1	Lig.2	Lig.2
Road						
Grass						
Tree						
Sky						

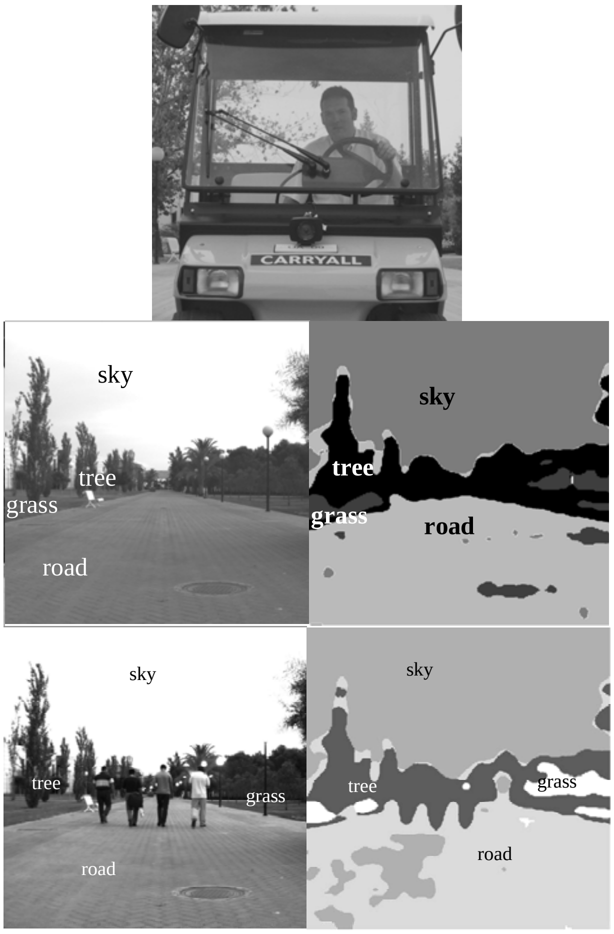


Figure 6. Vehicle from which the image sequences were captured and the results of surface labelling in diferent situations

5.2 Perfomance Tests

Since the main objective of this article is the performance of the system: clock cycles of the system modules, processing frequencies and sample rate of the images assuming the speeds of the vehicle taken as an example will be discussed in detail below.
Table 2 shows the estimate of the minimum duration of the clock (clk) for each module (logic and routed) and the resources used, provided by the tools of the Xilinx ISE 5.2i in the synthesis phase. Once the whole circuit has been mapped and routed, units are repeated and the routing becomes more difficult. This results in an increase in spatiotemporal costs. In addition, it must be taken into account that memory access is 17 ns on the prototype card, thus theoretically indicating a frequency of 58.82 MHz. However, in the performance tests performed directly on the card, the delays produced by the logic and router indicate a (clk) clock of 40 MHz.

Table 2. Estimate of the duration of the clock phase and area occupied by the FPGA.

Module	clk (MHz)	% Resources FPGA
Histogram	192.493	1.17
Normalization	115.674	4.67
SOM	99.572	72.71
UC	189.861	0.09
Histogram+SOM+UC	81.031	87.95

A comparative test has been made between the software simulation for AMD (Athlon 1.7 GHz) and Intel (Pentium IV 1.7 GHz) and the hardware prototype. In order to do this, the simulation has been modified to take into account the aspects of the implementation (Manhattan distance, 32 components and 400 neurons). Table 3 shows the frames/second that each of the alternatives for images of 1024 x 1024 pixels is capable of processing for different sample frequencies (distances in pixels between one sample window and the next). The prototype is able to calculate from 2.52 to 130.71 images per second.

Table 3. Comparison of images/second of the simulator and the hardware prototype

Sampling Distance. fm	Sampling Frequency v/img	AMD img/s	INTEL img/s	RC1000 img/s
4	62.001	0.05	0.05	2.52
8	15.625	0.20	0.17	9.94
12	7.000	0.49	0.45	21.76
16	3.969	0.91	0.83	37.86
20	2.560	1.46	1.42	57.14
24	1.792	2.07	2.02	78.06
28	1.327	2.82	2.78	103.19
32	1.024	3.56	3.80	130.71

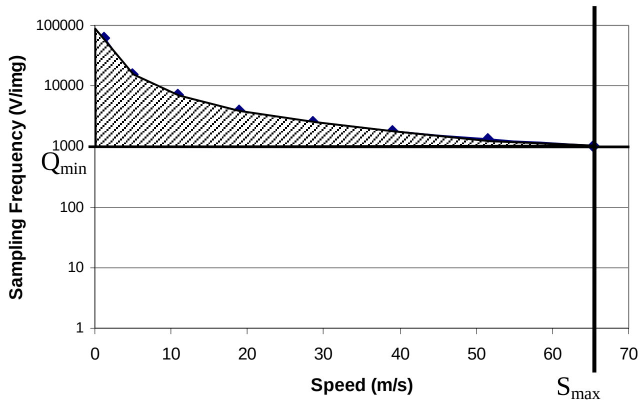


Figure 7. Relationship between the vehicle speed and the maximun number of sample widows suported by the system. Also we can see the maximun speed for a minimum quality established (Q_{min}).

In the real time tests, the security objective for the vehicle is to obtain a labelling result for a fixed portion of space (labelled image/meter). In this situation, for each value of vehicle speed we have a labelling frequency limitation. For example, if we want to obtain 2 labelled images each meter; for a vehicle speed value in meters/s, we would have to offer the double in labelling speed in frames/s. If the vehicle moves at 28 meters/s, we will have to label at least at the rate of 56 images labelled by second. In this case, we could choose sampling distances greater or equal to 20 (Table 2), being 20 (fm) the option of greater quality.

In figure 7 we show the relationship between the vehicle speed and the number of sample widows suported by the system, to achieve 2 labelled images each meter. The remarked area determines the possible combinations of speed and quality that can appear in the application over a minimum quality established (Q_{min}).

6 CONCLUSION

A system for real scene labelling has been presented. The system provides the labelling of unstructured scenes irrespective of the lighting and scale capture conditions and the frequencies required. The system is based on a model for vision in real conditions that stores descriptors of images perceived with successive values of the optical parameters, in this case, lighting. The model proposes a preprocessing phase in order to estimate the context and lighting conditions, followed by a processing phase where the architecture reconfiguration takes place and partial views of the databases can be queried. These databases are implemented by means of self-organizing maps that organize the brightness histograms extracted from the images per surface. The system is applied as a guidance aid for vehicles that leaves the specific use of the labelling provided (land maps, route planning …) open, and will depend on the type of application (moving vehicles, robots, aid to the visually handicapped …). A FPGA based prototype has been provided and tested in its use as a guidance aid for vehicles in outdoor scenarios. The processing frequencies obtained enable labelling results between sections in a reduced space to be obtained for the vehicle used in the experiments. This prototype can regulate its labelling frequency at the speeds demanded by the vehicle by means of changing the sample frequency and the consequent variation in quality response.

REFERENCES

[1] Flusser J., and Suk T.: Degraded Image Analysis: An Invariant Approach. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 6. June 1998.

[2]. Malik J., Belongie S., Leung T., and Shi J.: Contour and Texture Analysis for Image Segmentation. IJCV 43(1).2001.

[3]. Shanahan, J.G., Baldwin, J.F., Thomas, B.T., Martin, T.P., Campbell, N.W., and Mirmehdi., M.: Transitioning from Recognition to Understanding in Vision using Cartesian Granule Feature Models. Additive Proceedings of the Intn'l conf. of the NAFIPS. New York. 1999.

[4]. García, J.M., Fuster, A., Azorín, J. and Maciá, F.: Architecture for Image Labelling in Real Conditions. ICVS 2003. Lecture Notes In Computer Science. Springer-Verlag. 2003.

[5]. García, J.M., Fuster, A., Azorín, J.: Image Labelling in Real Conditions. Kybernetes (The International Journal of Systems & Cybernetics) Ed: MCB University Press. Vol. 34. No. 10. 2005.

[6]. Hadjidemetriou, E., Grossberg, M.D., Nayar, S.K.: Histogram Preserving Image Transformations. International Journal of Computer Vision 45(1). October 2001.

[7]. Strat, T., and Fischler, M.: The Role of Context in Computer Vision. In Proceedings of the ICCV. Workshop on Context-Based Vision.1995.

[8]. Kohonen, T.: Self-Organizing Maps. Springer-Verlag, Berlin Heidelberg, 1995.

[9]. Hammerstrom, D. and Nguyen, N.: An Implementation of Kohonen's Self-Organizing Map on the Adaptive Solutions Neurocomputer, in Artificial Neural Networks, T. Kohonen et al., (Eds.), Elsevier Science Publishers, pp. 715-719. 1991.

[10] Rückert, U.: ULSI Implementations for Artificial Neural Networks. 9th Euromicro Workshop on Parallel and Distributed Processing. 2001.

[11] Rüping, S., Porrrmann, M. and Rückert, U.: SOM accelerator system, Neurocomputing 21, pp. 31-50, 1998.

[12]. Ratha N.K. and Jain A.K.: Computer Vision Algorithms on Reconfigurable Logic Arrays. IEEE Transactions on Parallel and Distributed Systems, 10(1):29-43, 1999.

[13]. Benkrid, K., Sukhsawas, S., Crookes, D. and Benkrid, A.: An FPGA-Based Image Connected Component Labeller. FPL 2003, LNCS 2778, pp. 1012–1015, Springer-Verlag. 2003

[14] García, J.M., Fuster, A., Azorín, J.: Real environments image labelling based on reconfigurable architectures. Field-Programmable Logic and Applications, Proceedings Lecture Notes in Computer Science 3203: 1104-1106. 2004