A Visual Inspection System for Rail Detection & Tracking in Real Time Railway Maintenance

P. De Ruvo[1], G. De Ruvo[1], A. Distante[1], M. Nitti[1], E. Stella[1] and F. Marino[*,1,2]

[1]*Istituto di Studi sui Sistemi Intelligenti per l'Automazione CNR, 70126 Bari, Italy*

[2]*Dipartimento di Elettrotecnica ed Elettronica, Politecnico di Bari, 70125 Bari, Italy*

Abstract: Rail inspection is an essential task in railway maintenance and is periodically needed. Inspection is manually operated by trained human operator walking along the track searching for visual anomalies. This monitoring is unacceptable for slowness and lack of objectivity. This paper deals with a patented *V*isual *I*nspection *Sy*stem for *R*ailway maintenance, devoted to different tasks. Here, its Rail Detection & Tracking Block (RD&TB) is presented.

RD&TB detects and tracks, into the acquired video sequence the rail head, by this way, notably reducing the area to be analyzed and inspected by other modules of VISyR. Thanks to its hardware implementation, RD&TB performs its task in 5.71 μs with an accuracy of 98.5%, allowing an on-the-fly analysis of a video sequence acquired up at 190 km/h.

RD&TB is highly flexible and configurable, since it is based on classifiers that can be easily reconfigured in function of different type of rails.

Keywords: Infrastructure maintenance, artificial vision, computing architectures, FPGA, single value decomposition, principal component analysis.

INTRODUCTION

The railway maintenance is a particular application context in which the periodical surface inspection of the rolling plane is required in order to prevent any dangerous situation. Usually, this task is performed by trained personnel that, periodically, walks along the railway network searching for visual anomalies. Actually, this manual inspection is slow, laborious and potentially hazardous, and the results are strictly dependent on the capability of the observer to detect possible anomalies and to recognize critical situations.

With the growing of the high-speed railway traffic, companies over the world are interested to develop automatic inspection systems which are able to detect rail defects: these systems could both increase the ability in the inspection, and reduce the needed time, therefore allowing a more accurate and frequent maintenance of the railway network.

VISyR is a patented *V*isual *I*nspection System for *R*ailway maintenance, which has been already introduced in [1], and is able to perform different tasks. In this paper, we discuss its capability in detecting and tracking the rail head by means of a FPGA-based Rail Detection and Tracking Block (RD&TB), which automatically detects and tracks the rail-head center into a video sequence.

RD&TB is a strategic core of VISyR, since to detect the coordinates of the rail is fundamental in order to reduce the areas to be analyzed during the inspection. More in detail, RD&TB is based on the Principal Component Analysis (PCA). This approach has been chosen among others not only because of its high accuracy, but specially because it is able to detect the coordinates of the center of the rail by analyzing only a single row of the acquired video sequence. This feature extremely reduces the I/O bandwidth, which constitutes a bottle-neck in the performance of the system. Moreover, thanks to its FPGA-based implementation, RD&TB performs its task in 5.71 μs with an accuracy of 98.5%, allowing an on-the-fly analysis of a video sequence acquired up at 190 km/h.

The paper is organized as follows. In Section II, an overview of the system is presented. Rail Detection & Tracking methodologies are resumed in Section III. FPGA-based hardware implementation is described in Section IV. Experimental results and computing performance are reported in Section V. Finally, conclusive remarks and future perspectives are drawn in Section VI.

SYSTEM OVERVIEW

VISyR's acquisition system is installed under a diagnostic train during its maintenance route, and it is composed by a DALSA PIRANHA 2 line scan camera [2] having 1024 pixels of resolution (maximum line rate of 67 kLine/s), using the Cameralink protocol [3], and it is provided with a PC-CAMLINK frame grabber (Imaging Technology CORECO) [4]. An appropriate illumination setup equipped with six OSRAM 41850 FL light sources reduces the effects of variable natural lighting conditions and makes the system robust against changes in the natural illumination. Moreover, in order to synchronize data acquisition, the line scan camera is triggered by the wheel encoder. This trigger sets the resolution along the main motion direction y at 3 mm (independently from the train velocity). The pixel resolution along the orthogonal direction x is 1 mm.

*Address correspondence to this author at the Dipartimento di Elettrotecnica ed Elettronica, Politecnico di Bari; via Orabona 4, 70126 Bari, Italy; E-mail: f.marino@poliba.it

A top-level logical scheme of VISyR is represented in Fig. (**1**), while Fig. (**2**) reports a screenshot of VISyR's monitor.

RD&TB employs PCA followed by a Multilayer Perceptron Network Classification Block (MLPNCB) for computing the coordinates of the center of the rail. More in detail, a Sampling Block (SB) extracts a row of 800 pixels from the acquired video sequence and provides it to the PCA Block (PCAB). Firstly, a vector of 400 pixels, extracted from the above row and centered on x_C (i.e., the coordinate of the last detected center of the rail head) is multiplied by 12 different eigenvectors. These products generate 12 coefficients, which are fed into MLPNCB, which reveals if the processed segment is centered on the rail head. In that case, the value of x_C is updated with the coordinate of the center of the processed 400-pixels vector and online displayed (see Fig. (**2**)). Else, MLPNCB sends a feedback to PCAB, which iterates the process on another 400-pixels vector further extracted from the 800-pixel row.

The detected values of x_C are also fed back to various modules of the system, such as SB, which uses them in order to extract from the video sequence some windows of 400x128 pixels centered on the rail to be inspected by the Defect Analysis Block (DAB): DAB convolves these windows by four Gabor filters at four different orientations (Gabor Filters Block). Afterwards, it determines mean and variance of the obtained filter responses and uses them as features input to the SVM Classifier Block which produces the final report about the status of the rail. More details on the implemented methods were given in [5].

Fig. (**1**) also shows as x_C is fed back into the Prediction Algorithm Block (PAB) in order to extract, from the video sequence, the windows which are predicted to contain the bolts that fasten the rails to the sleepers. These windows are provided to BDB, which reveals presence/absence of these bolts. A more detailed description of PAB and BDB is given in [1].

PCAB, MLPNC and BDB are implemented in hardware on an Altera's StratixTM FPGA. SB, PAB and DAB are software tools developed in MS Visual C++ 6.0 on a Work Station equipped with an AMD Opteron 250 CPU at 2.4 GHz and 4 GB RAM.

RAIL DETECTION & TRACKING METHODOLOGIES

RD&TB is a strategic core of VISyR, since "to detect the coordinates of the rail" is fundamental in order to reduce the areas to be analyzed during the inspection. A rail tracking system should consider that:

- the rail may appear in different forms due to the type and wear of it. Each rail is classified by UIC [17] nor-

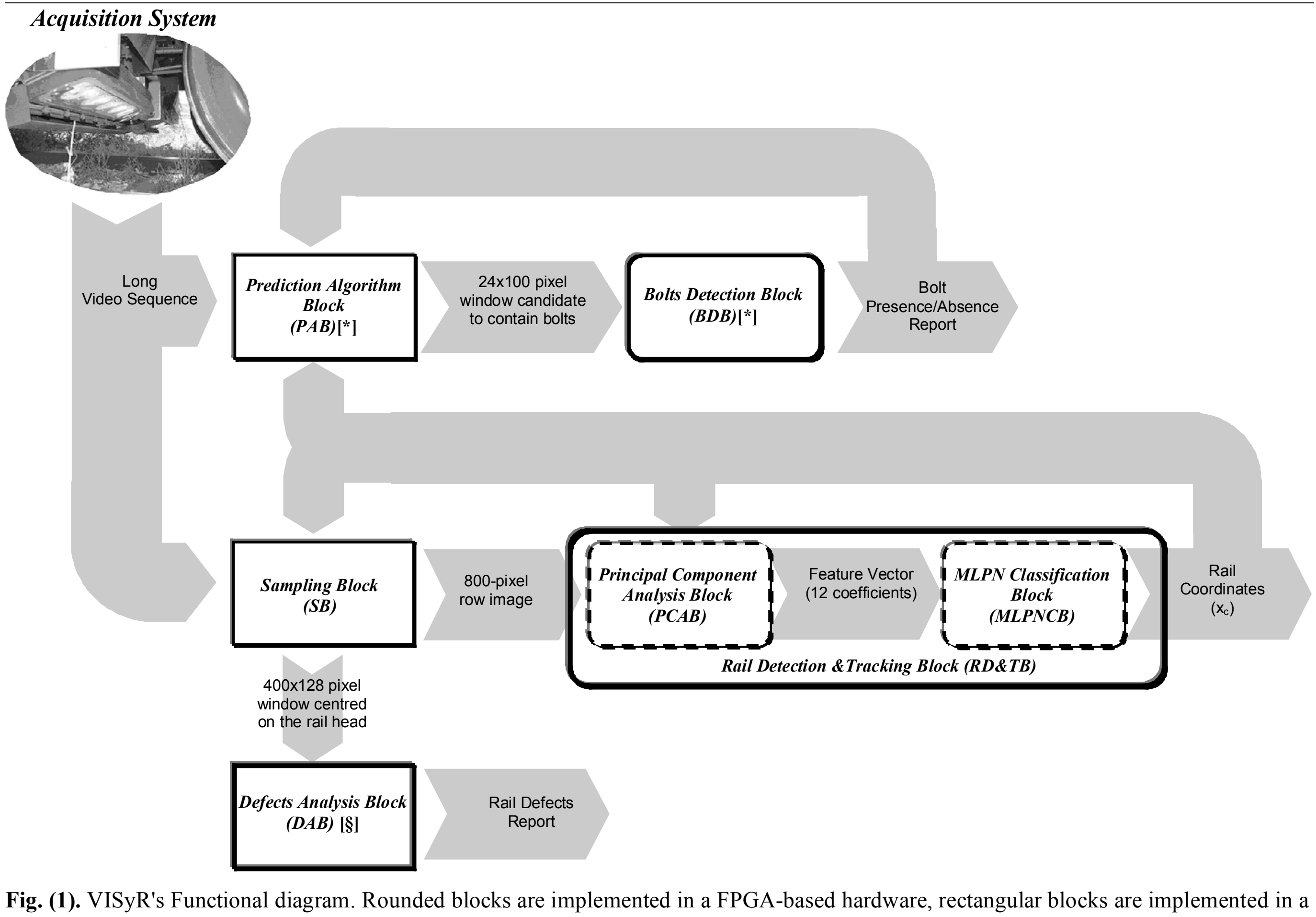

Acquisition System
Long Video Sequence
Prediction Algorithm Block (PAB)[*]
24x100 pixel window candidate to contain bolts
Bolts Detection Block (BDB)[*]
Bolt Presence/Absence Report
Sampling Block (SB)
800-pixel row image
Principal Component Analysis Block (PCAB)
Feature Vector (12 coefficients)
MLPN Classification Block (MLPNCB)
Rail Coordinates (x_c)
Rail Detection &Tracking Block (RD&TB)
400x128 pixel window centred on the rail head
Defects Analysis Block (DAB) [§]
Rail Defects Report

Fig. (1). VISyR's Functional diagram. Rounded blocks are implemented in a FPGA-based hardware, rectangular blocks are implemented in a software tool on a general purpose host. [*] and [§] denote blocks and methodologies respectively described in [1] and [5].

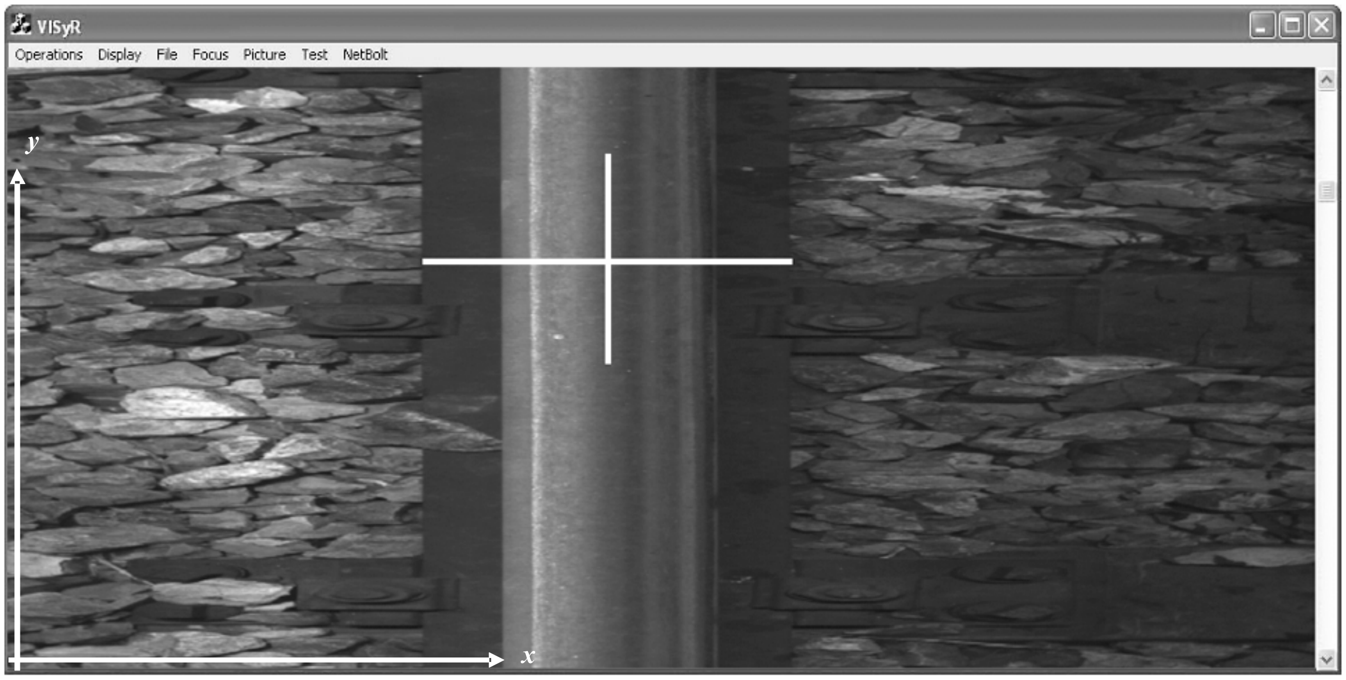


Fig. (2). VISyR's monitor. Rail Detection and Tracking.

normative. In Italy the rails frequently used are "UIC 50" and "UIC 60";

- the rail illumination might change;
- the defects of the rail surface might modify the rail geometry;
- in presence of switches, the system should correctly follow the principal rail.

In order to satisfy all of the above requirements, we have derived and tested different approaches, respectively based on Correlation, on Gradient based neural network, on Principal Component Analysis (PCA) with threshold and a PCA with neural network classifier.

Briefly, these methods extract a window ("patch") from the video sequence and decide if it is centred or not on the rail head. In case the "patch" appears as "centered on the rail head", its median coordinate x is assigned to the coordinate of the centre of the rail x_C, otherwise, the processing is iterated on a new patch, which is obtained shifting along x the former "patch".

Even having a high computational cost, PCA with neural network classifier outperformed other methods in terms of reliability. It is worth to note that VISyR's design, based on a FPGA implementation, makes affordable the computational cost required by this approach. Moreover, we have experienced that PCA with neural network classifier is the only method able to correctly perform its decision using as "patches" windows constituted by a single row of pixels. This circumstance is remarkable, since it makes the method strongly less dependent than the others from the I/O bandwidth. Consequently, we have embedded into VISyR a rail tracking algorithm based on PCA with MLPN classifier. This algorithm consists of two steps:

- a data reduction stage based on PCA, in which the intensities are mapped into a reduced suitable space (Component Space);
- a neural network-based supervised classification stage, for detecting the rail in the Component Space.

PCA-based Data Reduction Stage

Principal Component Analysis [6, 7] is a powerful tool for reducing the amount of data to be analysed, mapping them into a space having the highest variance.

Let us consider $\mathbf{i}_j$ row-images each one having only one row of N pixels, object of the analysis.

Let R a set of P images $\mathbf{r}_k$ (k=1..P, $P \geq Q$). Such images $\mathbf{r}_k$, having Q pixels with $Q < N$, have been extracted from the images $\mathbf{i}_j$, and chosen in order to select instances of the objects.

Let $\mathbf{A}$ the Q rows and P columns matrix:

$$\mathbf{A}=[\mathbf{h}_1, \ldots, \mathbf{h}_P] \quad (1)$$

with:

$$\mathbf{h}_k = \mathbf{r}_k - \boldsymbol{\mu} \quad (2)$$

where:

$$\boldsymbol{\mu} = [\mu_1, .., \mu_Q]^T \quad (3)$$

with μ_i denoting the average of i pixel intensities in the P set of images, e.g.

$$\mu i = \sum_{j=1}^{P} r_j^i \Big/ P$$

From $\mathbf{A}$, the covariance matrix:

$$\mathbf{C}=\mathbf{A}\mathbf{A}^T \quad (4)$$

can be built. The QxQ matrix $\boldsymbol{C}$ contains information about mutual relationships among rail images $\mathbf{r}_k$.

The eigenvectors $\mathbf{u}_j$ (j=1..Q) of $\mathbf{C}$ define a new reference space in which the variance among data is maximized. Moreover, an ordering relationship on $\mathbf{u}_j$ components can be induced sorting the eigenvectors $\mathbf{u}_j$ in such way that:

$$\lambda_q > \lambda_{q+1} \; (q=1, .., Q-1) \tag{5}$$

where the eigenvalues λ_j of $\mathbf{C}$, represent the variances of each one of $\mathbf{u}_j$. In other words, (5) means that the set of projections of the input data on $\mathbf{u}_q$ has variance higher than that one of the set of projections of the input data on $\mathbf{u}_{q+1}$.

By thresholding the eigenvalues λ_j it is possible to select the corresponding $L<Q$ eigenvectors sufficient enough to represent the biggest part of the informative content of the input data. Let λ_l (l=1..L, $L<Q$) the selected components, a generic vector $\mathbf{r}$' can be expressed by:

$$\mathbf{r}' \approx \sum_{l=1}^{L} a_l \mathbf{u}_l + \mathbf{\grave{\imath}}' \tag{6}$$

where $\boldsymbol{\mu}$' is the average vector of $\mathbf{r}$'. From a computational point of view the eigenvectors and eigenvalues of $\mathbf{C}$ can be estimated by a Single Value Decomposition (SVD) of matrix $\mathbf{A}$ where the coefficients a_l are evaluated by the inner product:

$$a_l = (\mathbf{r}'-\boldsymbol{\mu}')\mathbf{u}_l^{\mathrm{T}} \tag{7}$$

In this scenario, the vector

$$\mathbf{a}'=[a_1 ,...., a_L]^{\mathrm{T}} \tag{8}$$

can be considered a feature containing most of information content of $\mathbf{r}$'.

Due to the setup of VISyR's acquisition, the linescan TV camera digitizes lines of 1024 pixels. In order to detect the centre of the rail head, we discarded the border pixels, considering rows of only N =800 pixels. In the set-up employed during our experiments, rail having widths up to Q = 400 pixels have been encompassed.

Matrices $\mathbf{A}$ and $\mathbf{C}$ were derived according to equations (1) and (4), using 450 examples of vectors. We have selected L=12 for our purposes, after having verified that a component space of 12 eigenvectors and eigenvalues was sufficient to represent the 91% of information content of the input data.

Neural Network-Based Classification Stage

The rail detection stage consists of classifying the vector $\mathbf{a}$' -determined as shown in (8)- in order to discriminate if it derives from a vector $\mathbf{r}$' centred or not on the rail head. We have implemented this classification step using a Multi Layer Perceptron Neural (MLPN) Network Classifier, since:

- neural network classifiers have a key advantage over geometry-based techniques because they do not require a geometric model for the object representation [6];
- contrarily to the id-tree, neural networks have a topology very suitable for hardware implementation.

Among neural classifiers, we have chosen the MLP, after having experimented that they are more precise than their counterpart RBF in the considered application, and we have adopted a 12:8:1 MLPN constituted by three layers of neurons (input, hidden and output layer), respectively with 12 neurons $n_{1,m}$ (m=0..11) corresponding to the coefficients of $\mathbf{a}$'derived by $\mathbf{r}$' according to (7); 8 neurons $n_{2,k}$ (k=0..7):

$$n_{2,k} = f\left(bias_{1,k} + \sum_{m=0}^{11} w_{1,m,k} n_{1,m} \right) \tag{9}$$

and a unique neuron $n_{3,0}$ at the output layer (indicating a measure of confidence on the fact that the analyzed vector $\mathbf{r}$' is centered or not on the rail head):

$$n_{3,0} = f\left(bias_{2,0} + \sum_{k=0}^{7} w_{2,k,0} n_{2,k} \right) \tag{10}$$

In (9) and (10), the adopted activation function $f(x)$, having range]0, 1[, has been:

$$f(x) = \frac{1}{1+e^{-x}} \tag{11}$$

while the weights $w_{1,m,k}$ and $w_{2,k,0}$ have been solved using the Error Back Propagation algorithm with an adaptive learning rate [8] and a training set of more than 800 samples (see Section V).

Algorithm

The Rail Detection and Tracking Algorithm consists of determining which extracted vector r' is centred on the rail.

Instead of setting the classifier using a high threshold at the last level and halting the research as soon as a vector is classified as centred on the rail ("rail vector"), we have verified that better precision can be reached using a different approach.

We have chosen a relatively low threshold (=0.7). This threshold classifies as "rail vector" a relatively wide set of vectors $\mathbf{r}$', even when these ones are not effectively centred on the rail (though they contain it). By this way, in this approach, we halt the process not as soon as the first "rail vector" has been detected, but when, after having detected a certain number of contiguous "rail vectors", the classification detects a "no rail". At this point we select as true "rail vector" the median of this contiguous set. In other words, we accept as "rail vector" a relatively wide interval of contiguous vectors, and then select as x_C the median of such interval.

In order to speed-up the search process, we analyse each row of the image, starting from a vector $\mathbf{r}$' centered on the last detected coordinate of the rail centre x_C. This analysis is performed moving on left and on right with respect to this origin and classifying the vectors, until the begin (x_B) and the end (x_E) of the "rail vectors" interval are detected. The algorithm is proposed in Fig. (**3**).

FPGA-BASED HARDWARE IMPLEMENTATION

Today, programmable logics play a strategic role in many fields. In fact, in the last two decades, flexibility has been strongly required in order to meet the day-after-day shorter time-to-market. Moreover, FPGAs are generally the first devices to be implemented on the state-of-art silicon technology. Therefore, even if FPGAs were initially created for developing little glue-logic, they currently often represent the core of various systems in different fields (e.g., [9-14]).

In order to allow VISyR to get real time performance, we have directly implemented in hardware RD&TB. We have adopted as development platform Altera's PCI High-Speed Development Kit, Stratix™ Professional Edition, which, among other features [15], presents a Stratix™ EP1S60F1020C6 FPGA, 256-MByte PC333 DDR SDRAM, 32-bit or 64-bit PCI and 8/16-bit differential I/O up to 800 Mbps.

The software environment for friendly designing, simulating and testing is Altera's Quartus II™. Fig. (**4**) shows a window of Quartus II™ CAD tool displaying a top-level schematic of our design.

1) Input Interface

The PCI Interface (not explicitly shown in Fig. (**4**)) sends the input data to the INPUT_INTERFACE block, through DataIn[63..0]. INPUT_INTERFACE separates the input stage from the processing stage, mainly in order to make the processing stage synchronous and independent from delays that might occur during the PCI input. Moreover, it allows of working at a higher frequency (clkHW signal) than the I/O (clkPCI signal).

2) Double Way Sliding Memory

As soon as the 800 pixel row is received by INPUT_INTERFACE, it is forwarded to the DOUBLE_WAY_SLIDING_MEMORY, where it is duplicated into 2 shift registers. These shift registers slide in opposite way in order to detect both the end and the begin of the rail interval according to the search algorithm formalized in Fig. (**3**).

```
x_C = 512;       // presetting of the coordinate of the centre of the rail

do Start image sequence to End image sequence;
    set r' (400-pixel row) centered on x_C;
    do:
        determine a' (12 coefficients) from r'
        input a' to the classifier and classify r'
        set the new r' shifting 1-pixel-left the previous r'
    while(r' is classified as rail)
// exit from do-while means you have got the begin of the "rail vectors" interval
    x_B = median coordinate of r';
    r' (400-pixel row) centred on x_C;
    do:
        determine a' (12 coefficients) from r'
        input a' to the classifier and classify r'
        set the new r' shifting 1-pixel-right the previous r'
    while(r' is classified as rail)
// exit from do-while means you have got the end of the "rail vectors" interval
    x_E = median coordinate of r';
    output x_C = (x_B+x_E)/2;
```

Fig. (3). Algorithm for searching the rail center coordinates.

The architecture can be interpreted as a memory: The task starts when the host "writes" a 800-pixel row to be analyzed. In this stage, the host addresses two shift registers inside the DOUBLE_WAY_SLIDING_MEMORY (pin address[12..0]) and sends the 800 bytes *via* the input line DataIn[31..0] in form of 200 words of 32 bits.

As soon as the machine has completed his job, the output line irq signals that the results are ready. At this point, the host "reads" them addressing the FIFO memories inside the OUTPUT_INTERFACE.

Modules Functionalities

A more detailed description of the modules is provided in the follow. This section will be concluded describing how a merged implementation of RD&TB and BDB (see details in [1]) has been achieved in a single Stratix™ EP1S60F1020C6 FPGA.

For saving hardware resources and computing time, we have discarded the floating point processing mode and we have adopted fixed point precision[1].

By this way, DOUBLE_WAY_SLIDING_MEMORY:

- extracts **r**' according the policy of Fig. (**3**);
- partitions **r** in four segments of pixels and inputs them to PREPROCESSING_PCA in four trances *via* 100byte[799..0].

3) PCA Preprocessing

[1]Before designing the hardware blocks, we have tested in software different fixed point precisions. As a result of these experiments, we have verified that a set up with 12 bits for the eigenvectors coefficients and 28 bits for the weights of the classifier, allows an accuracy only 0.6% lower than that one achievable using floating point precision.

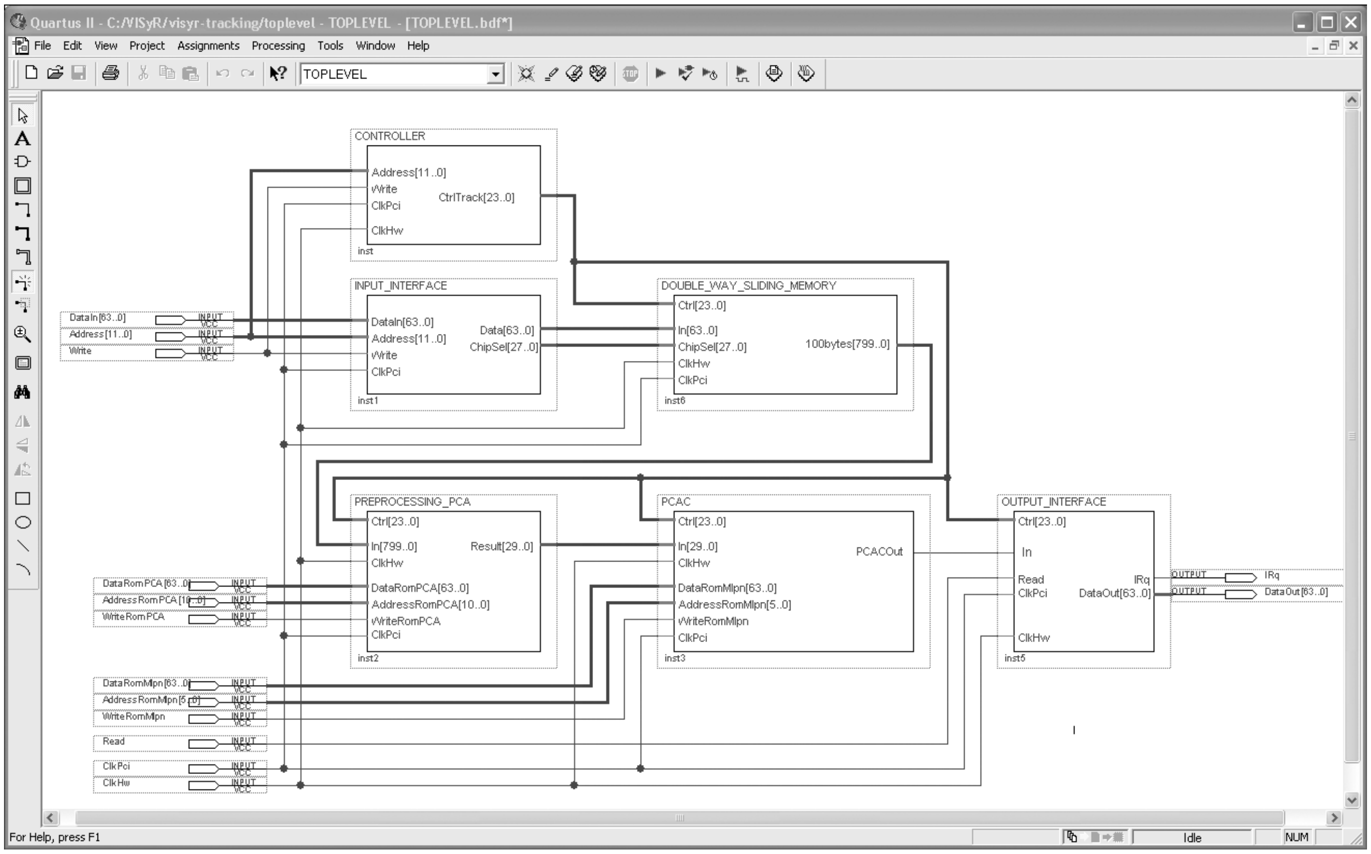


Fig. (4). A top-level schematic of the rail detection and tracking block, as it can be displayed on Altera's QuartusII™ CAD tool.

PREPROCESSING_PCA computes equation (7) in four steps. In order to do this, PREPROCESSING_PCA is provided with 100 multipliers, that in 12 clock cycles (ccs) multiply in parallel the 100 pixels (8 bits per pixel) of **r**' with 100 coefficients of $\mathbf{u}_m$(12 bits per coefficient, m=1..12). These products are combined in order to determine the 12 coefficients a_l (having 30 bits because of the growing dynamic) which can be sent to PCAC *via* Result[29..0] at the rate of 1 coefficient per cc.

This parallelism is the highest achievable with the hardware resources of our prototypal development board. Higher performance can be achieved with more performing devices.

4) Multi Layer Perceptron Neural Classifier

The results of PREPROCESSING_PCA has to be classified according to (9) and (10) by a MLPN classifier (PCAC).

Because of the high hardware cost needed for arithmetically implementing the activation function $f(x)$ -i.e., (11)-, PCAC divides the computation of a neuron into two steps to be performed with different approaches, as represented in Fig. (**5**).

Specifically, step (a):

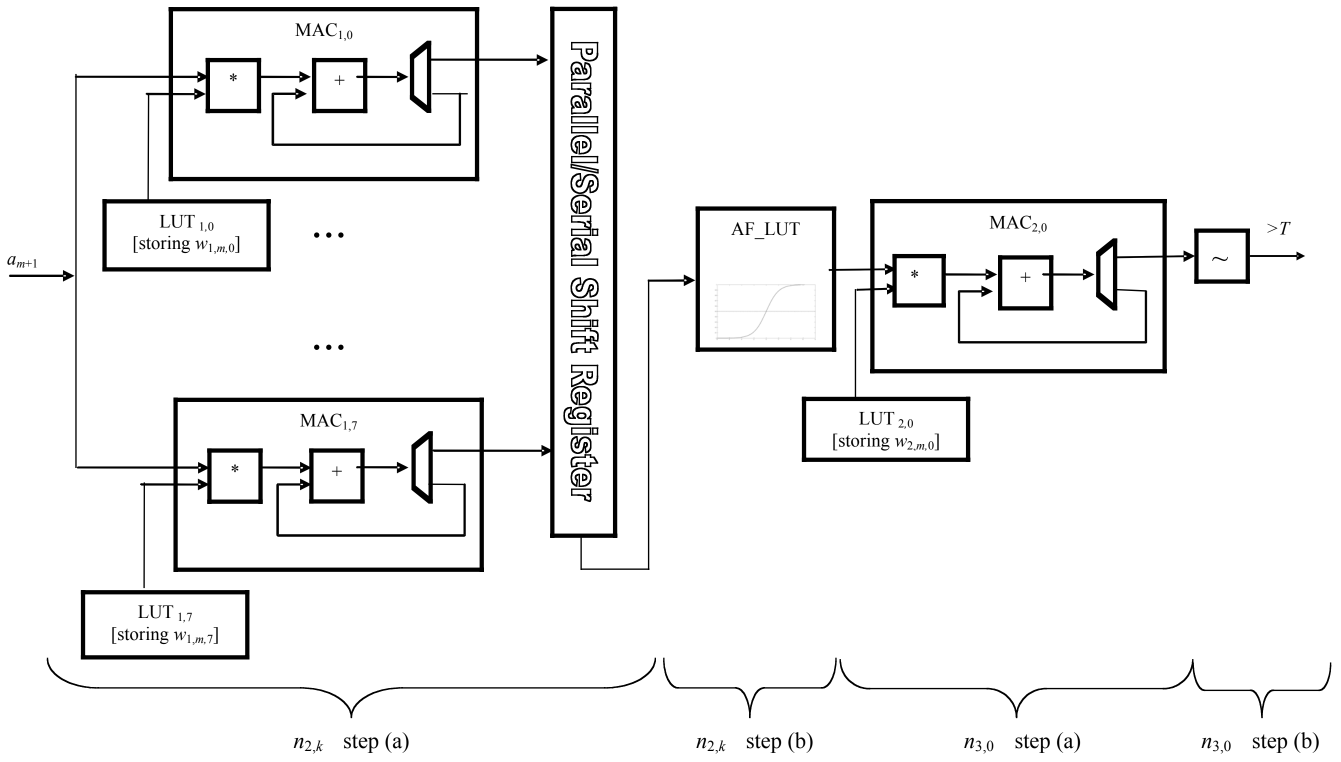
$$x = bias + \sum wn \tag{12}$$

is realized by means of Multiplier-and-ACcumulators (MACs), and step (b):

$$n = f(x) \tag{13}$$

is realized by means of a Look Up Table (for what concerns neurons $n_{2,k}$) and comparers (for what concerns neuron $n_{3,0}$).

- neurons $n_{2,k}$, step (a): PCAC has been provided with 8 Multiplier-and-ACcumulators (MACs), i.e., $\mathrm{MAC}_{1,k}$ (k=0..7), each one initialized with $bias_k$. As soon as a coefficient a_l (l=1..12) is produced by PREPROCESSING_PCA, the multipliers $\mathrm{MAC}_{1,\,k}$ multiply it in parallel by $w_{1,m,k}$ (m=l+1, k=0..7). These weights have been preloaded in 8 LUTs during the setup, $\mathrm{LUT}_{1,\,k}$ being related to $\mathrm{MAC}_{1,\,k}$ and storing 12 weights. The accumulation takes 12 ccs, one cc for each coefficient a_l coming from PREPROCESSING_PCA; at the end of the computation, any $\mathrm{MAC}_{1,\,k}$ have produced the value x_k.
- neurons $n_{2,k}$, step (b): The values x_k are provided as addresses to AF_LUT through a parallel input/serial output shift register. AF_LUT is a Look up Table which maps at any address x the value of the Activation Function $f(x)$. The adopted precision and sampling rate are discussed in Section V.
- neuron $n_{3,0}$, step (a): This step is similar to that of the previous layer, but it is performed using a unique $\mathrm{MAC}_{2,\,0}$ which multiplies $n_{2,k}$ (k=0..7) by the corresponding $w_{2,k,0}$ at the rate of 1 data/cc.
- neuron $n_{3,0}$, step (b): Since our attention is captured not by the effective value of $n_{3,0}$, but by the circum-



$MAC_{1,0}$
*
+
$LUT_{1,0}$ [storing $w_{1,m,0}$]
a_{m+1}
$MAC_{1,7}$
$LUT_{1,7}$ [storing $w_{1,m,7}$]
Parallel/Serial Shift Register
AF_LUT
$MAC_{2,0}$
$LUT_{2,0}$ [storing $w_{2,m,0}$]
~
$>T$
$n_{2,k}$ step (a)
$n_{2,k}$ step (b)
$n_{3,0}$ step (a)
$n_{3,0}$ step (b)

Fig. (5). PCAC functionality.

stance that this might be greater than a given threshold T=0.7 (the result of this comparison constitutes the response of the classification process), we implement step (b) simply by comparing the value accumulated by $MAC_{2,0}$ with $f^{-1}(T)$.

5) Output Interface

Because of its latency, PCAC classifies each pattern 5 ccs after the last coefficient is provided by PREPROCESSING_PCA. At this point, the single bit output from the comparer is sent to OUTPUT_INTERFACE *via* PCACOut.

This bit is used as a stop signal for two counters. Specifically, as soon as a value "1" is gotten on PCACOut, a first counter C_B is halted and its value is used for determining which position of the shift of the DOUBLE_WAY_SLIDING_MEMORY is that one centered at the begin of the "rail vector" interval. Afterward, as soon as a value "0" is received from PCACOut, a second counter C_E is halted signaling the end of the "rail vector" interval. At this point, Irq signals that the results are ready, and the values of C_B and C_E packed in a 64 bits word are sent on DataOut[63..0]. Finally, the host can require and receive these results (signal read).

Merged Implementation

The above architecture requires the resources summarized in Table **1**, which reports also a relative count with respect to the available resources on the Stratix™ EP1S60F1020C6 FPGA. These resources appear underutilized, since in the same FPGA is also embedded the BDB, which, as reported in [1] required the hardware resources that we report in Table **2**.

These data, put in evidence that the merged implementation in a unique board of RD&TB and BDB is not trivial, because of over utilized resources. Therefore, in order to achieve this result, we consider that according to VISyR's operating mode, bolt detection and rail tracking are not concurrent jobs. In fact, BDB reveals the presence/absence of bolts analyzing a restricted area of the acquired images, whose position is derived by the coordinates of the rail; coordinates that have been previously detected by RD&TB. This means that BDB and RD&TB work separately, and not simultaneously, inside the global architecture. Consequently, they can share some modules having a double functionality, i.e. performing tasks related to BDB in a time slice and tasks related to RD&TB in another time slice. In fact:

- the PCI Interface is independent from the blocks, meaning that we can implement only one PCI Interface, which identically works in all the time slices;
- INPUT_INTERFACE and OUTPUT_INTERFACE in RD&TB and BDB have some identical modules, meaning that merged version of these interfaces can be provided with only a copy of these modules;
- The 8 MAC_k (k=0..7) in Fig. (**5**) can be realized on 8 MACs among those implementing the Daubechies Classifier (DC) inside the block 1LEV_MLPN_CLASSIFIER of BDB (see [1] for details). This, simply coupling each one of these multipliers with two LUTs: the first one storing the neural weights of DC, and the second one storing those of PCAC. These

LUTs will be enabled alternatively, in order to provide the MAC with suitable the weights in the different time slices.

Because of these resource-optimizations, the global architecture can be designed as shown in Fig. (**6**), employing the resources summarized in Table **3**. Fig. (**7**) shows the generated floorplan for the Altera Stratix^{TM} EPIS60 FPGA. In this merged architecture:

- CONTROLLER, realizes the functionalities of the controllers of RD&TB and of BDB, through the buses CtrlTrack[23..0] and CtrlBolt[10..0] (in a single time slice, only one of these buses is active);
- INPUT_INTERFACE and OUTPUT_INTERFACE realize the merged functionalities of the homologous blocks of RD&TB and BDB;
- DOUBLE_WAY_SLIDING_MEMORY, and PRE-PROCESSING_PCA are the same as described here;
- HAAR_LL2_FILTER and DAUB_LL2_FILTER are the same as described in [1];
- CLASSIFIERS implement a shared DC/PCAC (as described above) and a Haar Classifier (HC): during the time slices devoted to RD&TB, DC/PCAC works as PCAC (HC is idle); whilst in the time slices devoted to BDB, it works as DC and HC works as described in [1]. Note that these blocks realize now the whole classification task (similarly as described in Section IV.1), and not only the first layer of the MLPNC, as written in [1], since that paper was written while the implementation of the second layer was a work in progress.

EXPERIMENTAL RESULTS AND PERFORMANCE

In order to design and test VISyR's processing core, a benchmark video sequence of more than 3,000,000 lines, covering a rail network of about 9 km was acquired. These were used in order to conduct several experiments aiming firstly at defining some methodological strategies and then at designing and testing the resulting system. In the following, several of the above experiments are described.

In Fig. (**8**) are reported the coordinates of x_C both real (i.e., manually extracted) and automatically estimated by the realized system. The average of the absolute error was 6.04 pixels. The only evident discontinuities occur in concomitance of three rail switches, resulting in the spikes of Fig. (**8b**) which reports the magnified error. We would put in evidence that, five other switches have been correctly analyzed. Anyway, except in these cases, the errors are almost always less than 10 pixels, and never more than 20. This error makes the method fully efficient for our practical purpose.

Single Value Decomposition Matrices Construction: Matrices **A** and **C** were derived according to (1) and (4) using 450 examples of vectors $\mathbf{r}_i$ extracted from the acquired video sequence. After having determined the eigenvectors $\mathbf{u}_j$ and their eigenvalues λ_j, we verified that 12 eigenvectors were enough to represent the 91% of the information content of input data.

Table 1. Rail Detection & Tracking Block: Required Hardware Resources

	Employed Resources	Available Resources	Utilization
Logic Elements	26,072	57,120	45.6%
DSP Blocks	8	18	44.4%
Memory Blocks (512 bits)[2]	22	574	3.8%
Memory Blocks (4K bits)	229	292	78.4%
Memory Blocks (MRAM)	1	6	16.7%
Total memory bits	374,756	5,215,104	7.2%
PLLs	1	12	8.3%
Total pins	168	782	21.5%

Table 2. Bolt Detection Block: Required Hardware Resources (as in [1])

	Employed Resources	Available Resources	Utilization
Logic Elements	32,465	57,120	56.8%
DSP Blocks	15	18	83.3%
Memory Blocks (512 bits)	12	574	2.1%
Memory Blocks (4K bits)	86	292	29.5%
Memory Blocks (512K bits)	1	6	16.7%
Total memory bits	339,889	5,215,104	6.5%
PLLs	1	12	8.3%
Total pins	168	782	21.5%

Table 3. RD&TB and BDB Merged Implementation: Required Hardware Resources

	Employed Resources	Available Resources	Utilization
Logic Elements	51,417	57,120	90.0%
DSP Blocks	15	18	83.3%
Memory Blocks (512 bits)	105	574	18.3%
Memory Blocks (4K bits)	292	292	100.0%
Memory Blocks (512K bits)	1	6	16.7%
Total memory bits	446,239	5,215,104	8.6%
PLLs	1	12	8.3%
Total pins	168	782	21.5%

[2]Altera's Stratix FPGAs are provided with memories of three different sizes: 512 bits (32 words x 18 bits), 4K bits (128 words x 36 bits) and MRAM (4096 words x 144 bits).

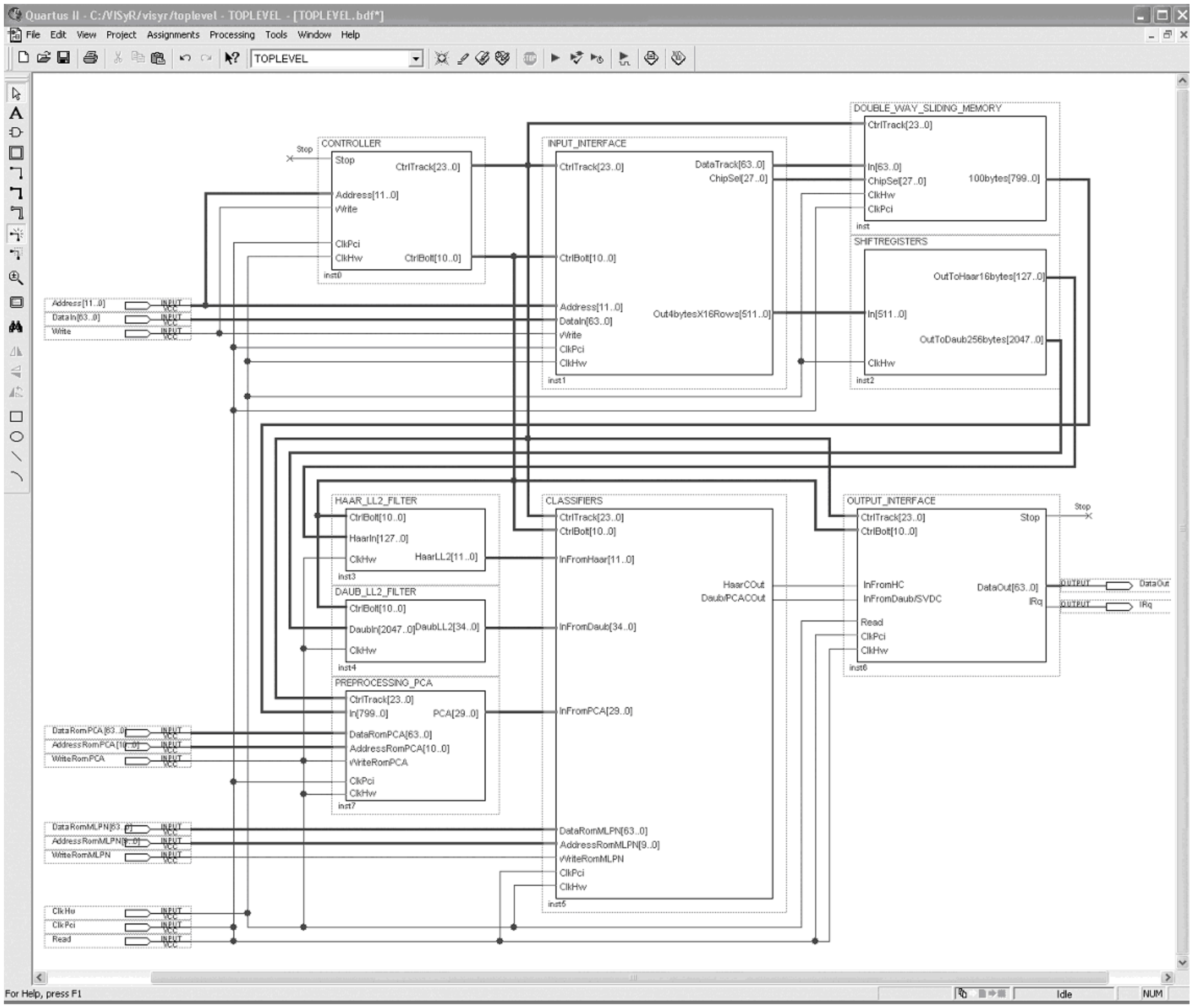

Quartus II - C:/VISyR/visyr/toplevel - TOPLEVEL - [TOPLEVEL.bdf*]
File Edit View Project Assignments Processing Tools Window Help
TOPLEVEL
CONTROLLER
INPUT_INTERFACE
DOUBLE_WAY_SLIDING_MEMORY
SHIFTREGISTERS
HAAR_LL2_FILTER
DAUB_LL2_FILTER
PREPROCESSING_PCA
CLASSIFIERS
OUTPUT_INTERFACE
For Help, press F1
Idle
NUM

Fig. (6). A top-level schematic of the merged implementation of RD&TB and BDB, as it can be displayed on Altera's QuartusII™ CAD tool.

MLPN Classifiers Training: Error Back Propagation algorithm with an adaptive learning rate [8] was used to determine the biases and the weights of the PCAC classifier. The adopted training set contained 262 different 400-pixels vectors centered on the rail (positive examples) and 570 negative examples consisting of 400-pixels vectors extracted from the video sequence.

Accuracy: The accuracy of the classifier was measured on a test set of more than 1,500 vectors (832 positives i.e., rails, 720 negatives i.e., non rails). 99.8% of positives and 98.2% of negatives were correctly detected. Activation Function Design: The analytical hardware implementation of the activation function f(x) -equation (3)- needs huge resources, as well as, introduces much latency. We have implemented it by a look up table AF_LUT, storing 4096 values f(x') computed onto 4096 equidistant values in [-5, 5] and assuming:

$$f(x)=\begin{cases}\text{if} \quad x<-5 & : \ 0\\ \text{if} \quad -5\le x\le 5 & : \ \text{AF\_LUT}[x']\\ \text{if} \quad x>5 & : \ 1\end{cases} \tag{14}$$

where x' is the rounded value of x.

AF_LUT was filled using words of 5 bits, that was found the best compromise in terms of detection accuracy and hardware cost.

CONCLUSION AND FUTURE WORK

This paper has proposed a Rail Detection & Tracking Block (RDT&B) which is part of VISyR, a patented Visual Inspection System for Railway maintenance, already introduced in [1]. RDT&B is based on PCA and automatically detects and tracks the center of the rail in a video sequence.

RD&TB has been fully implemented onto a FPGA, and holds a proper generality, its task being strategic in any vision inspecting system working in the field of railway maintenance. In fact, in this context, to extract portion of images centered on the rail is fundamental for strongly reducing the area to be analyzed and inspected in order to reveal any kind of anomalies. It reaches impressive performance, since it analyzes rows of video sequence in 5.71 μs, classifying them with an accuracy of 98.5%, reaching a velocity of 152 km/h, with peaks higher than 200 km/h. This performance takes into account that a significant part of the input stage cannot be overlapped (pipelined) with any other computation. Future research will deal with this bottleneck, for instance developing the FPGA-architecture on the same board where the frame grabber is located, avoiding the need for PCI input (e.g., ODYSSEY XPRO+, produced by MATROX and embedding both a FPGA and G4 PowerPC [16] and ANACONDA by CORECO IMAGING which embeds a FPGA and a PowerPC [4]).

The realized system is highly flexible and configurable, being its decision level based on classifiers that can be easily reconfigured in function of different type of rails to be detected and inspected. VISyR constitutes a significant aid to the personnel in the railway safety issue because of its high reliability, robustness and accuracy. Moreover, its computing performance allows a more frequent maintenance of the entire railway network.

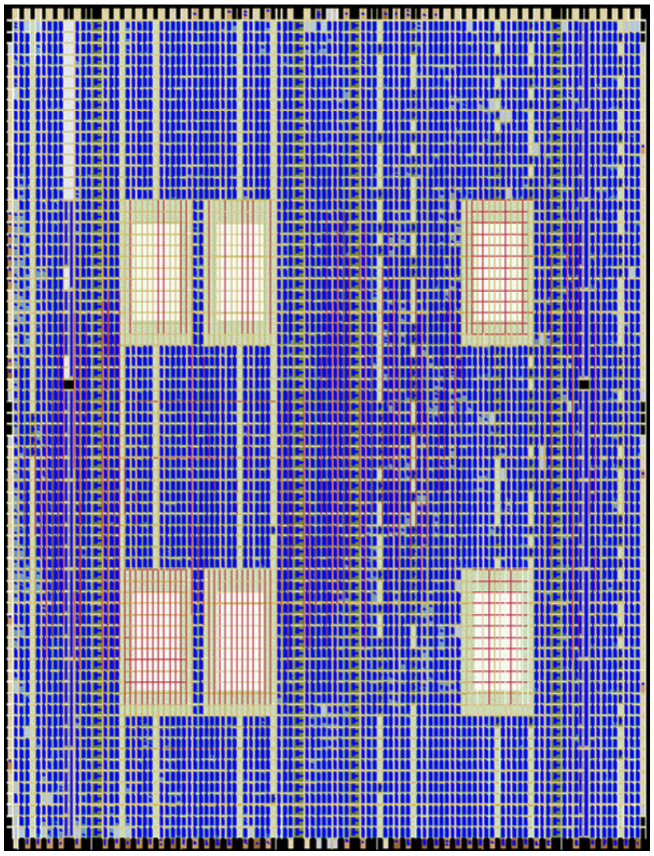
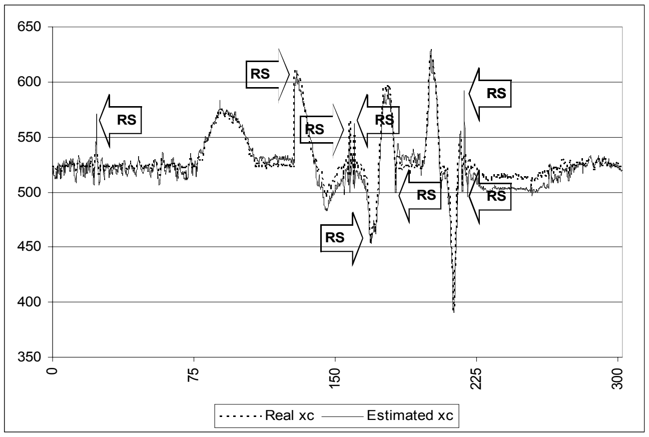


Fig. (7). Floorplan of Altera Stratix^{TM} EP1S60 after being configured.

(**a**)



(**b**)

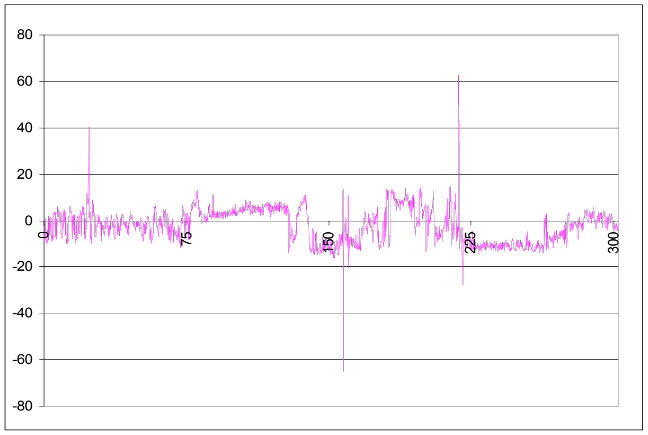


Fig. (8). (**a**): Real and estimated coordinates of x_C. (**b**): error. RS denotes rail switch.

ACKNOWLEDGEMENTS

This work has been partially supported by the Italian Ministry of University and Research, research project "RAILSAFE" MIUR- PON n° 12853.

REFERENCES

[1] F. Marino, A. Distante, P. L. Mazzeo and E. Stella, "A Real Time Visual Inspection System for Railway Maintenance: Automatic Hexagonal Headed Bolts Detection", *IEEE Transactions on Systems, Man and Cybernetics-Part C.* vol. 37, no. 3, May 2007, pp. 418-428.

[2] http://vfm.dalsa.com/products/features/piranha2.asp

[3] "CAMERALINK: specification for camera link interface standard for digital cameras and frame grabbers," www.machinevisiononline.org

[4] http://www.coreco.com

[5] C. Mandriota, M. Nitti, N. Ancona, E. Stella, A. Distante, "Filter based feature selection for rail defect detection, " *Machine Vision and Applications*, Springer-Verlag, vol. 15, pp. 179-185, Oct. 2004.

[6] A. Jain, R. Duin, & J. Mao "Statistical Pattern Recognition: A Review", IEEE Transactions on Pattern Analysis and Machine Intelligenve, vol. 22, no.1, pp.4-37, Jan. 2000.
[7] S. Gong, S. J. McKenna, and A. Psarrou. *Dynamic Vision: From Images to Face Recognition*, Imperial College Press. 2001.
[8] M. Bishop, *Neural Networks for Pattern Recognition*, New York, Oxford, pp. 164-191, 1995.
[9] R. Szplet, J. Kalisz, R. Szymanowski, "Interpolating time counter with 100 ps resolution on a single FPGA device," *IEEE Trans. Instrumentation and Measurement*, vol. 49, no. 4, pp. 879-883, Aug. 2000.
[10] S. Perri, P. Corsonello, M. A. Iachino, M. Lanuzza, G. Cocorullo, "Variable Precision Arithmetic Circuits for FPGA-Based Multimedia Processors," *IEEE Trans. Very Large Scale Integration Systems*, vol. 12, no. 9, pp. 995-999, Sept. 2004.
[11] B. A. Draper, J. R. Beveridge, A. P. W. Bohm, C. Ross, M. Chawathe, "Accelerated image processing on FPGAs," *IEEE Trans. Image Processing*, vol. 12, no. 12, pp. 1543-1551, Dec. 2003.
[12] C. M. Laymon, R. S. Miyaoka, B. K. Park, T. K. Lewellen, "Simplified FPGA-based data acquisition system for PET," *IEEE Trans. Nuclear Science*, vol. 50, no. 5, pp. 1483-1486, Oct. 2003.
[13] F. Rodriguez-Henriquez, N. A. Saqib, A. Diaz-Perez, "4.2 Gbit/s single-chip FPGA implementation of AES algorithm," *Electronics Letters*, vol. 39, no. 15, pp. 1115-1116, July 2003.
[14] C. Hinkelbein, A. Kugel, R. Manner, M. Muller, M. Sessler, H. Simmler, H. Singpiel, "Pattern recognition algorithms on FPGAs and CPUs for the ATLAS LVL2 trigger," *IEEE Trans. Nuclear Science*, vol. 47, no. 2, pp. 362-366, April 2000.
[15] http://www.altera.com/products/devkits/altera/kit-pci_stx_pro.html
[16] http://www.matrox.com/imaging/products/.
[17] UIC – Union Internationale des Chemins de fer – International Uinio of Railway http://www.uic.frodyssey_xcl/home.cfm.

Received: February 12, 2008 Revised: February 21, 2008 Accepted: February 22, 2008