VĚDECKÉ SPISY VYSOKÉHO UČENÍ TECHNICKÉHO V BRNĚ

Edice PhD Thesis, sv. 441

ISSN 1213-4198

Ing. Luděk Bryan

Hardware-Based Object Detection Method

BRNO UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology
Computer Systems

Ing. Luděk Bryan

Hardware-Based Object Detection Method

Hardwarově orientovaná metoda detekce objektů

Short version of Ph.D. Thesis

Study field:	Information Technology
Supervisor:	Doc. Ing. Vladimír Drábek, CSc.
Opponents:	Doc. RNDr. Elena Gramatová, CSc. Doc. Dr. Ing. Pavel Zemčík
Presentation date:	26. 11. 2007

Key Words

Object detection, hardware, FPGA, computer vision, template matching, reconfiguration, license plate

Klíčová slova

Detekce objektů, hardware, FPGA, počítačové vidění, porovnávání vzorů, rekonfigurace, státní poznávací značka

© Luděk Bryan, 2008
ISBN 978-80-214-3569-8
ISSN 1213-4198

OBSAH

1 INTRODUCTION

Forsyth and Ponce [17] defined the goal of computer vision "... to model and automate the process of visual recognition, a term we interpret broadly as perceiving distinctions between objects with important differences between them." Computer vision has been an important part of world-wide research since the 1970s, when computers started to be capable of processing large amounts of data. The computer vision field may serve for many purposes, including surveillance, robot control, autonomous vehicle driving, image set organization, scene analysis, face detection and many others.

Another relatively new area of computer science is programmable hardware (PLDs), which deals with hardware circuits capable of changing their internal structure, even during circuit operation. PLDs are largely used in *embedded systems*.

The goal of this thesis is to link these three fields - computer vision, embedded systems and programmable hardware, by suggesting a new method for object detection and designed for programmable hardware implementation.

1.1 WHY PLD IMPLEMENTATION IN EMBEDDED SYSTEMS?

Software solutions for computer vision are today very well explored. However, a different situation is in the field of VLSI design. This area is now quickly growing as new devices are on the market, both powerful and reasonably priced. Mainly PLD devices are now affordable even for small businesses.

An important feature of PLD chips is the possibility to be reprogrammed by the user. This feature, called reconfiguration, can be used not only for debugging and standard operation, but also for some more advanced operations. Mainly *dynamic reconfiguration* allows the user to quickly reprogram a design (or even a part of one) on a chip. Thus, completely new computer-related methods are being developed such as evolutionary computation, and evolvable hardware.

Embedded systems are becoming part of many things of everyday use, including cars, and home appliances. Making a design for an embedded system is comparable to designing a standard system. The biggest difference is a design space limited by price, size, and power consumption. It makes sense though to create the method suitable for the embedded systems, which significantly extends possible range of target applications.

To summarize, as new technologies have developed, a gap has risen in the field of computer vision techniques targeting PLDs, while this area is becoming more important as small businesses can use powerful PLDs for their embedded systems.

1.2 WHY DO WE NEED HARDWARE METHODS?

As stated by Leibson [18] the majority of programers today are used to sequential thinking. This was mostly caused by the dominating position of purely sequential computers for a long period of time. As predicted by Liebson, this habit has to change as the multi-core processors and PLD chips are getting a bigger share of the market.

With the mentioned considerations, one question naturally comes to mind: What are the advantages of hardware design (either general VLSI design, or specifically PLD design) over standard processor programming?

- The design can be highly parallel.
- Comparing to parallel processor or processor arrays, implementation of any arbitrary parallel algorithms is straightforward, not limited by the processor instruction set.
- Real-time designs are suitable.
- Custom behaviour of inputs / outputs is natural.
- Lower power consumption can be reached. Govindu et al. [19] compared CPU and PLD implementation of the same algorithm and the PLD implementation was nearly 8 times more energy efficient.
- For PLDs, chip reconfiguration is possible.

Of course, there are also disadvantages of hardware design:

- Implementation phase is difficult and time consuming.
- Sequential behaviour is expensive and difficult to implement.

As a result, it's not possible to say universally whether the processor programming or hardware design is better. Some tasks are suitable for hardware implementation, while others are more suitable for the processor. Usually, the most efficient way is by using either the processor, for naturally sequential or easy tasks, and parallel processors or a hardware custom design, together with a processor, for difficult tasks.

1.3 WHAT IS A METHOD DESIGNED FOR HARDWARE?

There are many methods that can be efficiently *implemented* in hardware. This thesis will go a little deeper and will also distinguish methods *designed* for hardware. What should be the features of such a method?

- Intended for hardware implementation
- Naturally parallel
- Not efficient software implementation
- Quick or immediate reaction required (software implementation may suffer from interrupt delays)
- Use of specific hardware features, like reconfiguration for PLD chips
- A method designed for hardware is more of an abstract idea, the listed items are more a clue than a definition.

2 PATTERN RECOGNITION

Pattern recognition is a computer vision technique for extracting high-level information from an image. This can be useful for object detection, object tracking, defect detection or robotic systems.

Pattern recognition is usually decomposed into three step pipeline - *preprocessing*, *feature extraction* and *classification*. Each of these is described in the following sections. However, in reality, many modifications of this pipeline exist, or some of the steps may be dissolved.

Preprocessing

Preprocessing enhances the image for easier processing during the next steps. This involves mainly removing noise from an image, but also much more sophisticated methods can be utilized. A list of feasible methods follows, with a brief description for each of them.

- *Point operators* are applied individually to every pixel in an image, without information about its neighborhood. Processing cost is usually low in comparison with other methods, but utilization is limited. Point operators include *thresholding*, *contrast stretching*, *linear*, or *non-linear operators*.
- *Spatial filters,* unlike point operators, transform a pixel according to its neighborhood, not only according to the pixel itself.
- *Histogram techniques* modify an image according to the changed shape of its histogram, or use histogram information for setting values for further processing.
- *Transform operations*, including the well-known Fourier or wavelet transform, do not just modify the image as previous methods do, but convert it to completely different form. Although transforms can be very powerful, implementation in hardware is very space consuming and difficult to implement.

Feature Extraction

The second step is *feature extraction* and is the essential part of pattern recognition. The goal is to recognize features in an image that indicate the presence of regions of interest. In other words, we want to transform an image from a spatial domain to a feature domain, where data represent more abstract quantities. Usually, the features are placed in the *feature vector* (v_1, v_2, ... , v_n).

If we use feature extraction for object detection or recognition, it should treat objects independently without regard to their placement or other properties. Usually, three characteristics are desirable:

- *Resistance against changes in illumination* is usually required. We should propose algorithms sturdy enough to handle shifts in intensity. Otherwise, our system is limited to exact lighting conditions.
- *Rotational invariance* is necessary when the objects in an image or sensor are not constantly oriented.
- *Scale invariance* means that the algorithm should work, if possible, equally well for closer or more distant objects.

For feature extraction, methods have been previously suggested by many authors, dependent on the target application [20], some of them in hardware [21]. The spatial filters are one of the more popular methods. An edge detector is an example of commonly used method, because edges are usually an important clue for pattern recognition.

Classification

The last step, known as *Classification*, decides whether there are any regions of interest, and where, upon the found features.

Classification can be either *supervised* or *unsupervised*. For supervised classification, a training data set is provided and learning has to be performed prior to classification. In unsupervised classification, the classifier decides about types of regions in an image solely from the information included in this image. In the thesis, we deal only with the supervised classification.

The fundamental idea of classification is to gather certain *features* from an image during the feature extraction phase, and decide what class the image belongs to.

There are several classification methods. One of the simplest methods is the *minimum distance classifier*. Each class is defined by one point in the n-dimensional feature space. A feature vector belongs to the closest class.

Bayesian decision theory is considered universal statistical method for classification. This method has been thoroughly described in many books, for example by Duda et al. [22]. *Neural networks* are also often used as a classification technique.

3 GOALS

Now the vital question comes: What should be the *new* feature of the proposed method, so that the proposed method is not only a parallelized version of a software method? The answer lies in the nature of hardware implementation, which allows massive parallel processing. However, for massive parallel processing the basic computation unit should be as simple as possible in order to place a large number of them into the target device. So the final goal is to implement a method that will consist of very simple basic elements, and the strength of the method will be in the implementation of a large number of those elements. Then, the basic hypothesis of this thesis is:

The proposed method, though based on very simple elements, can compete with commercially used methods due to the massive parallelization of those simple elements.

Summarized, the goal is to implement an object detection method with these features:

- *Method is designed for hardware* as stated in Section 2.3.
- *Simple basic building blocks* will be used in order to use massive parallelization.
- *Real-time processing* is required so that a solution for every image is found in a constant maximum time limit. If possible, computation should be done "on-the-fly", i.e. the data will be processed while being received at the input, and output is available after certain constant time delay. The "on-the-fly" feature is not necessary for real-time hardware implementation, but it brings some advantages like minimum delay or no need of buffer implementation.
- *Limited resources* are in an embedded system, we can not use a high-power PC processor or large memory blocks.
- *Adaptability to changes* in the environment using PLD reconfiguration should be possible.
- *Comparable results* to commercially used methods are required.

4 TEMPLATE BASED DETECTION METHOD

In Section 3 , the expected functionality of the proposed method is suggested. Coming from these expectations, a new method is proposed in this section.

The proposed method together with experimental architecture was described in [1] and [2].

We will go through a few standard object detection techniques, in order to decide which one will be used for the proposed method.

The method should be suitable for FPGA implementation and should profit from the advantages of FPGAs. This is not the case for any frequency domain transforms, Hough transform, thinning, or motion detection techniques. Edge detection is suitable for an FPGA implementation, but is a rather simple method and can be efficiently implemented in a DSP processor. Histogram techniques are suitable for an FPGA implementation and profit from hardware implementation. However, histogramming usually does not give sufficient results for the method as a whole, but can be used as a part of the method for segmentation. Template matching, is suitable for hardware implementation in non-linear filter form, and it can also benefit from hardware implementation by utilizing massive parallelization in the case of larger number of filters. Therefore, the template matching will be our object detection method.

In the basic form, where an object also represents a template, template matching may suffer from some important problems:

- The object will not look the same every time. I.e. we do not want to search for one object, but for certain class of objects.
- The object may consist of some repeating patterns. Having only one template for each of these repeating objects may significantly save implementation resources.
- There may be parts of an object not holding any information. However, appropriate detection hardware must be still implemented.

These issues may be solved by more sophisticated detection methods. However, the requirement from Section 3 is to suggest very simple detecting elements.

The basic idea on how to resolve this is to *decompose the searched object into smaller objects* (templates), *and search for each of these templates separately*, used by Qiang and Bo [24], for example. Presence of an object is then determined by dependencies of these templates. This way we can allow higher miss rate of the simple elements, while the object as a whole should still be detected. By this decomposition, we also will switch from searching for *instances of an object* to searching for *objects belonging to a certain class*. This feature is caused by the possibility of locating objects that do not exactly match a template.

The proposed method follows the standard three step pattern recognition scheme - preprocessing, feature extraction, and classification.

We want to do as much work in preprocessing as possible, which means leaving only the necessary part for feature extraction. This necessary part is comparing 1-bit values in templates with 1-bit values derived from an image, saying whether the pixel is dark or bright. This implementation will be very cheap in hardware - if templates are known at time of synthesis, it would require only invertors for dark pixels and one AND gate through all values (chip occupancy will be discussed more in Section 5.1).

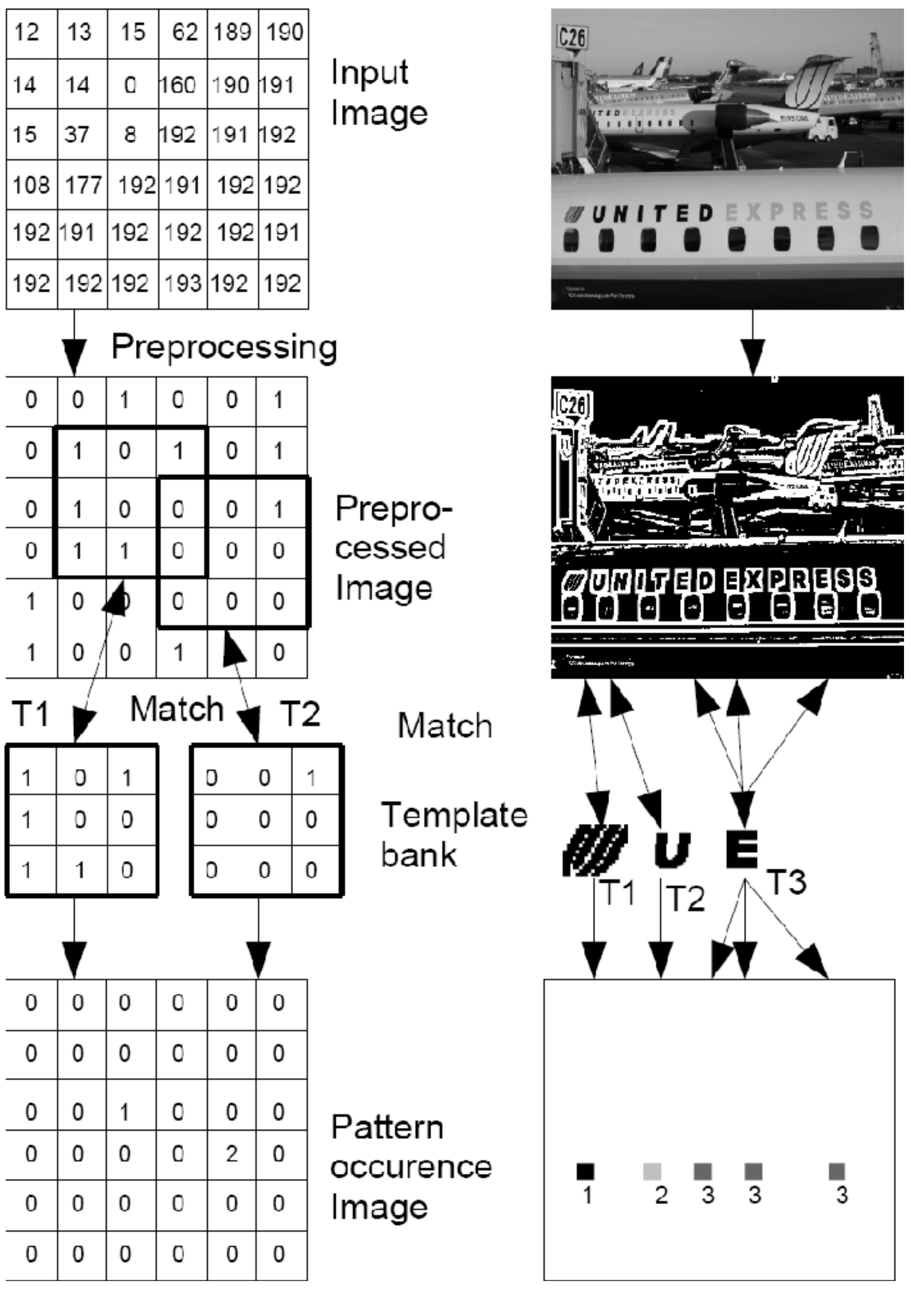
4.1 PREPROCESSING

The preprocessing unit has to convert an image to a binary image. How this conversion is done is largely application specific. In this thesis, preprocessing will be made suitable for the case study of the license plate detection. The whole preprocessing operation is implemented as a set of two filters - the edge detection filter and thresholding filter.

For the edge detection filter (1st stage), the output of the filter (pixel in the output image) will be the response of the edge filter. Many methods could be used for edge detection. The most common approach would be the linear filters. But, these filters are not suitable for hardware implementation. As an alternative, a 3x3 non-linear filter was developed. The function of this filter is the difference between the minimum and maximum of the pixels in a neighborhood.

The second stage is actual segmentation. A non-linear 3x3 filter computes a value from the neighborhood, and outputs 1 if the pixel value is greater than this value, or 0 otherwise. For the dividing value, average between minimum and maximum of the neighborhood pixels is used.

The preprocessing method proved to be very effective in preserving shapes and removing noise, which are basic considerations for the quality of feature extraction. An Example of a preprocessed image is in Figure 1.



12	13	15	62	189	190
14	14	0	160	190	191
15	37	8	192	191	192
108	177	192	191	192	192
192	191	192	192	192	191
192	192	192	193	192	192

0	0	1	0	0	1
0	1	0	1	0	1
0	1	0	0	0	1
0	1	1	0	0	0
1	0	0	0	0	0
1	0	0	1		0

1	0	1
1	0	0
1	1	0

0	0	1
0	0	0
0	0	0

0	0	0	0	0	0
0	0	0	0	0	0
0	0	1	0	0	0
0	0	0	0	2	0
0	0	0	0	0	0
0	0	0	0	0	0

Figure 1: Example of preprocessing and feature extraction

4.2 FEATURE EXTRACTION

For feature extraction, we need to utilize many parallel filters, each searching for one specific template. A template is represented as a binary image. The set of these templates will be called a *template bank*. The feature extraction filter puts identification numbers of a template in the template bank to template occurrence image to positions where corresponding templates fit.

An illustration of preprocessing and feature extraction is shown in Figure 1. On the right and left side real-life image and an artificially made close-up image are shown, respectively. In the feature extraction phase, we are searching for templates T_1 .. T_n included in the template bank. Finally, the pattern occurrence image is created with the located slices matching templates. In the right image, three occurrences of template "E"' (template T_3) have been found.

Unlike in the example, templates in a real system are expected to be smaller, automatically generated and their number will be in hundreds.

4.3 CLASSIFICATION

In this thesis, there will be no straight answer for how to implement the classification part, because it largely depends on the application. For our purposes, we will use two extreme versions of classification:

- *Minimal classification* where object is detected when the number of matched templates exceeds a certain number in an area of the objects' size
- *Maximal classification* where object is detected when there are slices matching templates at the exact position as on the searched object

In reality, we will probably use something between those two. For instance, specific slices have to be placed in certain areas.

5 EXPERIMENTAL ARCHITECTURE

Figure 2 shows the overall scheme. The connecting line descriptors show the format of the data. Input to the system is a serial stream of data with bit length *D* representing pixels coming from a sensor. The pixels are ordered from left to right, lines from top to bottom. The whole scheme works "on-the-fly", i.e. whenever a new pixel comes to the input, a new pixel is computed and appears at the output (with a certain delay, caused by the serial to matrix unit).

Block *serial to matrix* (published in [6])is necessary, because filters work with surrounding of the actual pixel. It converts the serial pixel input to a matrix of NxN pixels, that contains the actual pixel and its surroundings. For this purpose, N-1 image lines must be stored in memory. For FPGA implementation, the most suitable are internal Block RAMs.

With this matrix of pixels (NxN pixels with bit depth D), we can do the preprocessing in the *edge detection* and *threshold* units.

After preprocessing, feature extraction can take place using a template bank. It compares all |B| templates with an image slice coming from the threshold unit, and if some of the templates fit, the proper output signal is set. There is a maximum of |B| output signals, one for each template. However, some of the templates can share output signals which reduces the number of signals. Then, the appropriate signal is set when any of the joint templates match.

An improvement was suggested by placing the templates directly into the processing elements of the target FPGA, because the chip occupation is critically dependent on the implementation of this unit.

The output image should be an array with the identification numbers of matched templates, so the set of |B| signals must be converted to a number of the matched template in *arr2num* unit. This number is finally the serial output forming a template occurrence image.

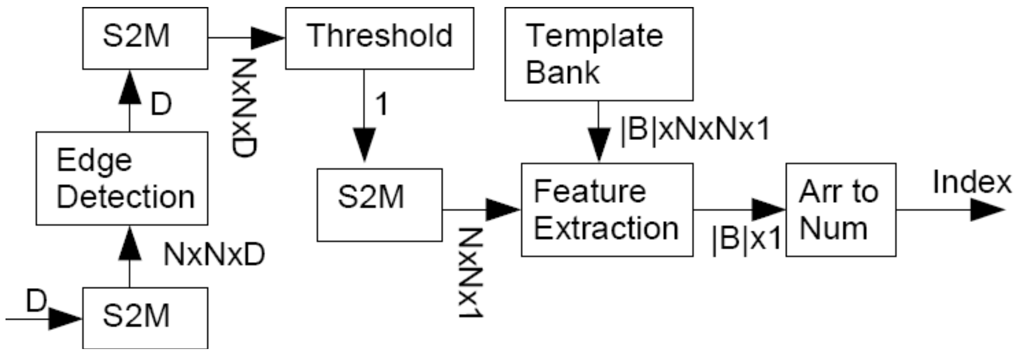


Figure 2: Overall hardware scheme

5.1 OVERALL SYNTHESIS RESULTS

In this section, the synthesis results of all the units are presented. The target FPGA is the Xilinx Spartan III XC3S1600E.

	Slices	BRAM	Max. delay
Edge detection	468	2 kB	21.0 ns
Thresholding	352	2 kB	14.4 ns
Feature extraction	4537	0	18.31 ns
XC3S1600E	14752	648 kB	N/A
Total	5357 [36%]	4 [1%]	21.0 ns

Table 1: Overall synthesis results

The results are shown in

Table 1. The design occupies 36% of the XC3S1600E FPGA chip. Maximum propagation delay of 21 ns allows “on-the-fly” implementation, as a normal HDTV cameras' pixel clock is usually 40 ns.

These results show that the method can be implemented in a real system, in fact there is no need of the more expensive, cutting-edge technology, FPGAs.

6 ADAPTIVE TEMPLATES

The dynamic reconfiguration block scheme is in Figure 3. It is a modification of the hardware scheme in Figure 2. Together with the normal feature extraction process (using "current" template bank) there is a parallel branch for the feature extraction of new templates being tested ("test bank"). The results from both of these branches are evaluated and compared in the processor. The testing branch creates new templates for the test bank, and removes the worst templates. The normal branch adds good templates to the current bank from the test bank.

In order to replace old templates with new ones using FPGA dynamic reconfiguration, the templates have to be in fixed predefined positions. Probably the easiest way to do this is to place the templates in a regular array directly into the target technology cells, for Xilinx FPGAs these are LUTs. This allows us to change the template pixels simply by reconfiguring only the LUT configuration. A description of the the dynamic reconfiguration approach was published in [3].

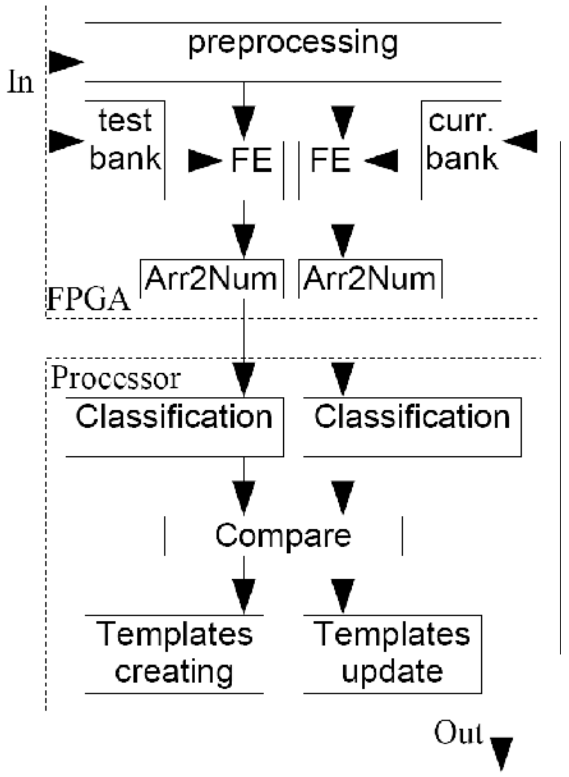


Figure 3: Dynamic reconfiguration

7 THE PROPOSED METHOD APPLICATION

To demonstrate that the proposed method suggested in Section 4 can be competitive with methods currently used in real life, the proposed method had to be implemented in an actual real-life situation and compared with some existing methods. This happened to be a difficult task, as the real-world methods and the testing data sets are usually proprietary. Fortunately, there was a chance to test the method on the real-life problem of detecting the license plates, the Unicam system[1] [4]. In this section, the proposed method will be evaluated and compared to the method based on a DSP processor architecture currently used in the Unicam system.

Implementation of the preprocessing and feature extraction is completely the same as proposed in Section 4 . The classification part differs, which will be described in Section 7.3.

7.1 PREPROCESSING

An Example of an original image and a preprocessed image is in Figure 4.

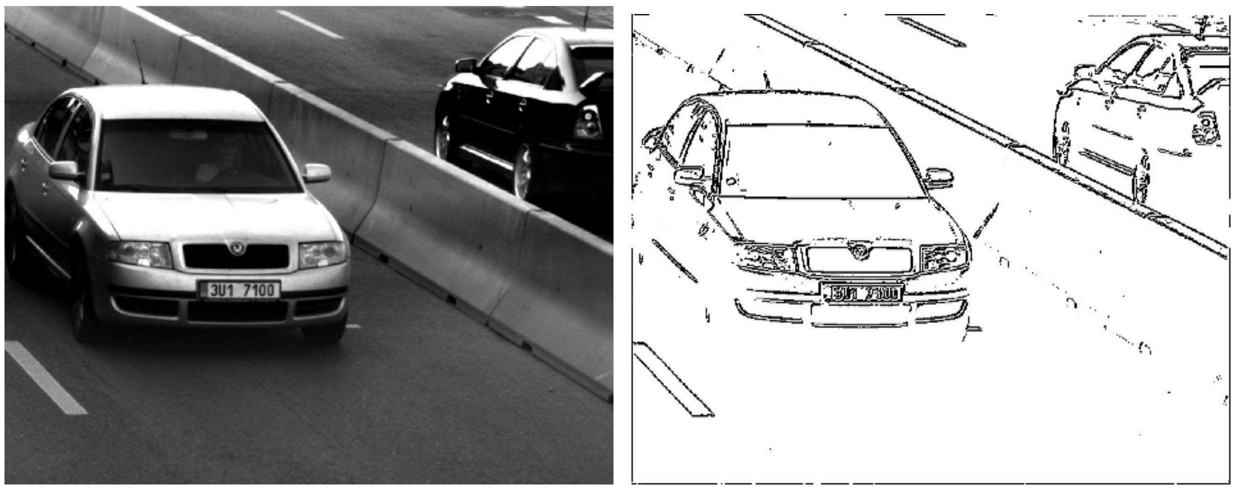


Figure 4: Original image and preprocessing

7.2 FEATURE EXTRACTION

Using the theoretical outputs regarding detection quality, a 5x5 template size has been chosen and the number of templates is 500. An example of a subset of an automatically created template bank is shown in Figure 5. An example of an image after feature extraction is in Figure 6, preprocessed image is printed light and matched templates dark.

[1] Thanks is given to Camea ltd. for supporting this research by providing me with the hardware boards and sample images. However, none of the outputs of the research part of this thesis have been used for commercial purposes by Camea ltd.

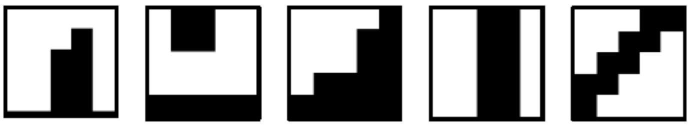


Figure 5: A template bank subset

7.3 CLASSIFICATION

The task of classification is to find the license plate, if present, in an image. A successful classification example is shown in Figure 6, the rectangle shows the detected license plate.

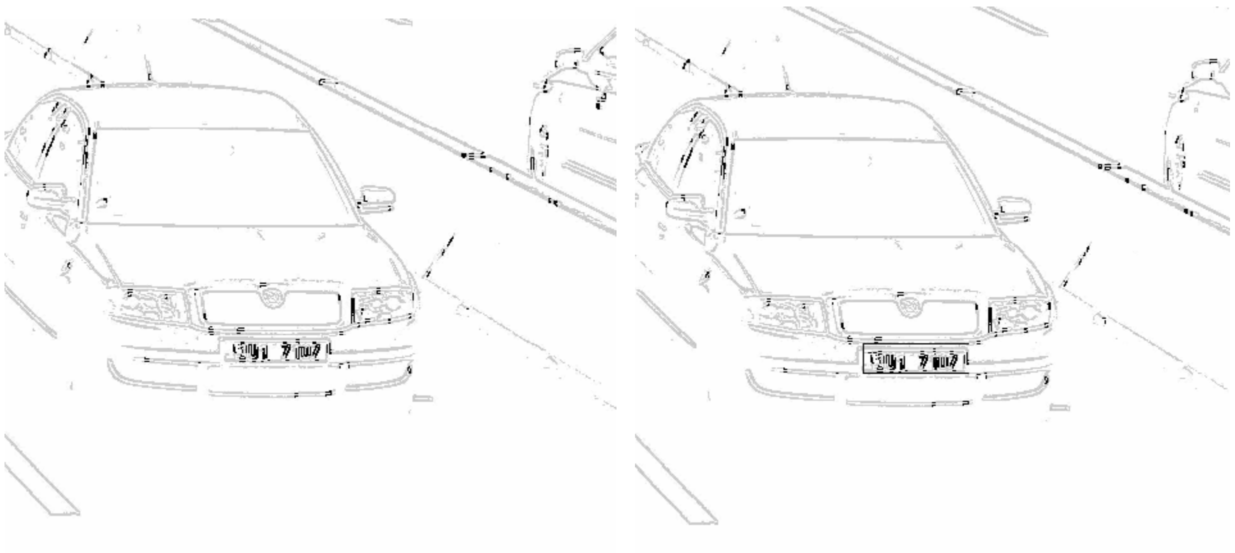


Figure 6: Feature extraction and classification

Classification, considering only the number of templates in an object, suffers significantly from the regular patterns outside of the license plate. For example, if a template detects the pattern of a vent (Figure 7), there may be more detected templates there, than in the license plate. However, although the number of templates is very high, there are usually only one or two templates involved.

The problem of regular patterns and other classification problems lead us to a modification of the classification function. One possibility is to consider not only the number of templates, but also how various they are (i.e. how many different templates are in the area).

The regular pattern problem seen in the last example shows that this case study is very important part of this thesis. If the proposed method testing was performed only on a limited number of images, the results would not have told us much about the actual method's properties.

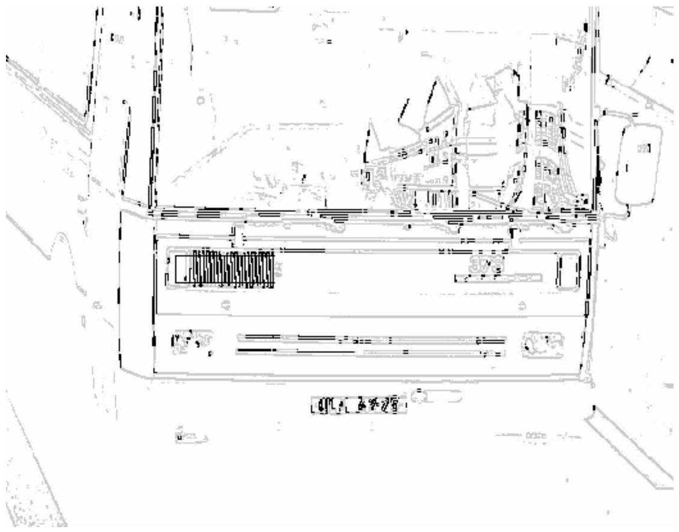


Figure 7: Regular pattern problem

7.4 EXPERIMENTAL RESULTS

The method was tested on a set of images from real traffic. The results are compared to the method that is currently used for detection in Unicam cameras, and has been developed for many years. For evaluation, the metrics suggested by Mariano et al. [23] seem to be suitable. However, there is no data available concerning the accuracy of detection for the currently used method. As a result, only the binary information, whether the object is detected or not, was used.

The proposed method has not been implemented as a whole in hardware yet. All tests were performed by a C program that works the same way as the design works in hardware.

The method needs to be trained first for every class of images by creating an appropriate template bank. For every image set, a subset containing 10 to 30 images was selected. Using a C program, 500 templates were obtained that were used as a bank for the whole image set.

	Current	Proposed	Set	Occ
Normal	94.1%	97.1%	377	33%
Light	96.7%	100%	29	3%
Dark	62.5%	93.6%	63	3%
Shadows	90.2%	78.4%	162	1%
Road	100%	100%	233	51%
Snow road	99.1%	99.7%	321	9%
Tilted images	40.4%	91.8 %	98	N/A
Total	96.6%	98.6%	1283	100%

Table 2: Experimental results

The results are presented in

Table 2. *Current* and *proposed* are the two compared methods, *set* is the number of images in each testing set, and the *occ* column is the rough estimate of percentages showing how often each group takes place in real-life. The results are shown in percents of successful detections. The last line, *total*, shows an estimate of the overall hit rate using the *occ* column.

The proposed method hit rate shows 2% better results than the current one. The credibility of this result may be affected by the limited test set, estimation of the *occ* values and the need of training set for each testing image group. However, the results still show that the proposed method quality is at least comparable to the current one.

7.5 SPEED UP AND PRICE

In this section, the proposed method is compared to two other implementations. First, with the implementation of the current method used in Section 7.4, implemented on the embedded processor. This comparison is the most important, because the current method serve as a reference. Second, we compare the proposed method implemented in C, on a PC computer, and in hardware to show the speed up achieved by moving the algorithm to hardware.

7.5.1 Comparison to the Current Method

The proposed method computes outputs “on-the-fly”. Considering an image slice of size of 860x105 and a maximum propagation delay 21.0 ns, total computation time is

$$T_{hw} = 860 \times 105 \times 21.0 \text{ ns} = 1.9 \text{ ms}$$

The current method runtime on the embedded processor TMS 320C6416 is on average T_{dsp} = 30 ms, depending on the image complexity. The algorithm has been optimized for parallel processing in the processor. The speed up of the proposed method is

$$S_{dsp} = T_{dsp} / T_{hw} = 30 / 1.9 = 15.8$$

Even though the hardware and software solutions work differently (dedicated hardware can not be used for other purposes while not in use), the speedup of 15.8 shows that the hardware solution is at least comparable to the DSP one.

The price of the suggested FPGA chip, the Spartan-3E 1600, is around \$63. The price of the DSP chip, the TMS 320C6416, is around \$150. The fraction of the hardware solution price over the DSP solution price is

$$P_{dsp} = \$63 / \$150 = 0.42$$

A simple price comparison is not going to give us a precise answer about which solution is cheaper. With the hardware solution, we still need to use some kind of processor for subsidiary functions. Also the other way around, with the DSP solution, we may need an FPGA chip for the low-level system functions. Still, the chip prices give us pretty good look on the final solution price.

7.5.2 Comparison to Software Implementation

The proposed method running on the Pentium M 1.6 GHz processor takes $T_{pm} = 9350$ ms. The speed up of the proposed method running on an FPGA is

$$S_{pm} = T_{pm} / T_{hw} = 9350 / 1.9 = 4921$$

This speed up shows how efficient FPGA implementation may be for highly parallel tasks compared to processor implementation. Not only is the speed up itself extensive, but an FPGA runs at 48 MHz while the processor runs at 1.6 GHz.

However, these numbers do not reflect any real-life situation. The speedup is more or less proof that it makes no sense to implement the method in software, and that the method is suitable for hardware implementation. Considering statements from Section 1.3, the speed up helps to realize one of the goals that the method was designed for hardware implementation.

8 CONCLUSIONS

A new method for object detection has been proposed. An important feature of the proposed method is that it has been *designed for hardware implementation*. The proposed method is based on very simple elements, filters detecting only one template each. The filters are implemented in a parallel matrix in an FPGA. Although based on simple elements, the proposed method's results are comparable to commercial method results.

8.1 THE PROPOSED METHOD FEATURES

The proposed method follows the standard three step pattern recognition scheme - preprocessing, feature extraction, and classification.

Preprocessing consists of two filters - an edge detection filter and a local thresholding filter. A binary preprocessed image is passed to the feature extraction stage, which consists of a large set of filters that run in parallel. Each of these filters compares an image slice to a template. A template is basically a small binary image representing a small part of an object. The set of these templates is called a template bank. The final phase of the object detection process is classification. For classification, various standard techniques can be used depending on the application that the method is being used for.

8.2 EXPERIMENTAL RESULTS

To show that the proposed method can be used in a real-life system, a case study dealing with the license plate detection was utilized. The proposed method was evaluated and compared to the method based on DSP processor architecture currently used in the Unicam system in Section 7.4. The evaluation was made on a set of almost 1300 real-life images from different environments. The results show that the detection quality the proposed method is at least comparable to the current one.

In Section 7.5.1, the proposed method and the DSP Unicam method hardware implementations were compared. The speed up of the proposed method against the current one is 15.8, and the solution of the proposed method using an FPGA seems to be approximately twice cheaper than the current DSP based solution.

In Section 7.5.2, the proposed method was compared to the software implementation of the same method. A speed up of about 5000 shows that the proposed method was designed for hardware, as discussed in Section 1.3.

8.3 GOAL FULFILLMENT

The requirements specified in Section 3 have all been met, namely:

- *The method is designed for hardware*, as the core of the method, feature extraction, is feasible for FPGA implementation. The proposed method can also profit from reconfiguration as shown in Section 6 .
- *Simple building blocks* are used. Template matching units are very simple, and can be placed in only a few FPGA slices.
- *Real-time processing* is used. Feature extraction is computed "on-the-fly"
- *Limited resources* are considered for method implementation. A standard FPGA is sufficient for the hardware part. The processor part, classification, is not too computationally difficult, thus an embedded processor implementation is suitable.
- *Adaptability to changes* is a part of the method. The technique proposed in Section 6 allows adaptation to the environment or slow object changes.
- The *results* in Section 7.4 show that the method efficiency is comparable to commercial methods.

8.4 ORIGINAL CONTRIBUTION

- I suggested a set of mathematical definitions for working with basic computer vision operations.
- In Section 4 , I proposed a new method for object detection designed for hardware implementation.
- I analyzed the proposed method and suggested optimal size of the templates. I defined image quality for the method and image class quality.

- In Section 5 , I proposed a hardware scheme for the proposed method, and confirmed its workability by simulation and synthesis of all of the major parts.
- In Section 6 , I extended the proposed method to adaptation to environmental changes using partial dynamic FPGA reconfiguration.
- In Section 7 , I applied the proposed method to the real-world problem of a license plate detection. To set the method parameters properly, I used a theory based on object detection quality.
- In Section 7.4, I compared the detection abilities of the proposed method to a real-world license plate detection method. The testing set was composed of nearly 1300 real-world images.
- In Section 7.5, I compared the speed up and price of the proposed method to a real-world license plate detection method.
- I confirmed the proposed method's ability to detect different types of objects by applying the method to another case study.
- I explored the proposed method's limits by setting the learning set and counter set to objects that are very similar.

8.5 FUTURE RESEARCH

Even though the method has been tested on real data and all parts have been synthesized, there is a long way to real implementation. Consequently, the first future goal is to make the system work in real hardware system, processing real data on site. A statistical evaluation of the results of this on-site system will help to find possible weak spots in the method, resulting in making the method more efficient.

Having the real hardware system would also allow us to analyze some other method's features. For example, in Section 1.2, we stated that a PLD implementation can be significantly more energy efficient than a processor implementation. If this statement was confirmed, it would be an important advantage of the proposed method.

9 SHRNUTÍ

V této práci je navržena nová hardwarová metoda detekce objektů v obraze. Základem této metody je paralelní extrakce příznaků pomocí jednoduchého porovnávání vzorů. Díky paralelismu je metoda vhodná pro hardwarovou implementaci. Součást práce je také experimentální architektura, založená na implementaci v FPGA. Navržená metoda může být rozšířena o adaptivní učící se systém, který automaticky nastavuje parametry v závislosti na okolním prostředí. V práci je navržen tento adaptivní systém pomocí dynamické rekonfigurace FPGA. Na ukázku toho, že navržená metoda může být použitá pro řešení problémů v reálném prostředí, byla navržena a vyhodnocena případová studie zabývající se detekcí státní poznávací značky v obraze.

9.1 VLASTNOSTI NAVRŽENÉ METODY

Navržená metoda se skládá ze tří částí známých z teorie rozpoznávání vzorů – předzpracování, extrakce vlastností a klasifikace.

Předzpracování je navrženo jako dva filtry – filtr detekce hran a filtr lokálního prahování, které vytvoří binární obraz ze vstupního obrazu ve stupnici šedé. Tento binární obraz je následně zpracován pomocí techniky extrakce vlastností, kde je obraz filtrován pomocí množství paralelních filtrů. Každý z těchto filtrů porovnává okolí v obraze s určitým vzorem. Tyto vzory jsou binární obrazy o rozměru v řádu jednotek až desítek pixelů, které reprezentují části hledaného objektu. Závěrečná fáze detekce je klasifikace, kde mohou být použity různé standardní techniky.

9.2 EXPERIMENTÁLNÍ ARCHITEKTURA

Pro potvrzení toho, že metodu je možné implementovat v praxi, pro ni byla navržena experimentální architektura. Jako cílová platforma byl zvolen FPGA čip ze standardní řady Spartan-3 firmy Xilinx. Všechny důležité jednotky navržené experimentální architektury byly syntezovány, díky čemuž bylo možné přibližně určit obsazení FPGA čipu a maximální frekvenci hodin. Díky optimalizaci modulu extrakce vlastností (umístění filtrů přímo do logických buněk cílové architektury) je možné metodu implementovat bez problémů do standardního FPGA.

9.3 ADAPTIVNÍ VZORY

V rámci práce byla také navržena technika, která umožňuje automaticky adaptovat vzory pro extrakci vlastností v závislosti na okolních podmínkách. K dosažení tohoto cíle je využito dynamické rekonfigurace FPGA.

Algoritmus se snaží získat nové vzory z detekovaných objektů při běžném provozu zařízení. Nové vzory jsou testovány, a pokud jsou vyhodnoceny jako kvalitní, jsou přesunuty do skupiny běžně používaných vzorů. K tomuto je potřeba mít možnost zapisovat vzory do filtrů FPGA za běhu zařízení.

Základem techniky je umístění filtrů extrakce vlastností přímo do logických buněk cílové architektury (LUT), jak již bylo naznačeno v sekci 9.2. Pixely vzoru, což jsou zároveň parametry filtru, jsou umístěny jako konfigurace buňky LUT. Jednotlivé filtry jsou umístěny pravidelně v poli buněk LUT. Díky tomu je možné změnit pixely vzoru pouhým přepsáním odpovídající buňky LUT. Pozici jednotlivých filtrů není těžké zjistit díky tomu, že buňky jsou umístěny v FPGA pravidelně v poli.

9.4 EXPERIMENTÁLNÍ VÝSLEDKY

Pro ověření, že navržená metoda může být použitá v praxi, byla provedena případová studie na reálném problému detekce státních poznávacích značek. Výsledky dosažené navrženou metodou byly porovnány s výsledky metody, která se používá v reálném systému Unicam. Testovací sada obsahovala téměř 1300 obrazů z reálného provozu. Výsledek testování ukázal, že navržená metoda dosahuje porovnatelných, nebo lepších výsledků než stávající metoda. Navržená metoda také vykazuje zrychlení 15.8 oproti stávající metodě, a odhadnutá cena implementace je zhruba poloviční.

9.5 SPLNĚNÍ CÍLŮ

Při návrhu metody byly stanoveny požadavky, které by metoda měla splňovat. Následuje jejich seznam s krátkým vysvětlením.

- *Metoda je navržená pro hardware.* Jádro metody, extrakce vlastností, je složená z velkého množství paralelních filtrů. Metoda také využívá možností rekonfigurace FPGA.
- *Jednoduché základní komponenty* jsou filtry, lehce implementovatelné v hardware. Každý z filtrů zabere pouze několik buněk LUT.
- *Zpracování v reálném čase* je realizováno díky implementaci “on-the-fly”, což znamená, že výsledky se objevují na výstupu se stejnou frekvencí jako data přicházející na vstup, pouze s určitým konstantním zpožděním.
- *Implementace na systému s omezenými zdroji* je možná. Výsledky zpracování experimentální architektury ukázaly, že pro implementaci je dostatečné standardní FPGA. Díky tomu je metoda vhodná i pro implementaci ve vestavěných systémech.
- *Přizpůsobení se změnám prostředí* je popsáno v sekci 9.3.
- *Výsledky srovnatelné se systémy používanými v praxi* jsou popsány v sekci 9.4.

REFERENCES

Author’s publications

[1] Bryan L., Fučík O., Drábek V.: HW-Based Object Detection Method for Traffic Monitoring, 6th Electronic Circuits and Systems Conference, Bratislava, SK, 2007

[2] Crha L.: System for the license plate detection and image compression using hardware, In: Proc. of the 7th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems, Bratislava, SK, SAV, 2004, p. 274-276, ISBN 80-969117-9-1

[3] Bryan L., Fučík O.: FPGA Implementation of a Reconfigurable License Plate Detection Method, Proceedings of the 2007 Engineering of Reconfigurable Systems and Algorithms, Las Vegas, NV, US, 2007

[4] Fučík O., Zemčík P., Tupec P., Crha L., Herout A.: The Networked Photo-Enforcement and Traffic Monitoring System Unicam, Proceedings of Engineering of Computer-Based Systems, Los Alamitos, IEEE, 2004, p. 423-428, ISBN 0-7695-2125

[5] Crha L., Fučík O., Šustek J.: Environment for Hw/Sw Codesign of Embedded Systems, Proceedings of the 8th IEEE Design and Diagnostic of Electronic Circuits and Systems Workshop, Sopron, HU, UWH, 2005, ISBN 9639364487

[6] Crha L., Fučík O., Drábek V.: Image filter implementation in FPGA used for the license plate detection, Proceedings of 38th International Conference Modeling and Simulation of Systems, 2004, Ostrava, CZ, MARQ, 2004, ISBN 80-85988-98-4

[7] Marek T., Novotný M., Crha L.: Design and Implementation of the Memory Scheduler for the FPGA - Based Router, Proc. of the Field Programmable Logic and Application 2004, Leuven, BE, Springer, 2004, p. 1133-1139, ISBN 3-540-22989-2

[8] Crha L.: Nové metody komprese, Sborník příspěvků ze semináře Počítačové Architektury & Diagnostika, Brno, CZ, FIT VUT, 2003, ISBN 80-214-2471-0

[9] Crha L.: Systém pro aplikačně specifickou kompresi obrazu, Zborník príspevkov Česko-slovenského seminára pre študentov doktorandského štúdia Počítačové architektúry & Diagnostika, Bratislava, SK, SAV, 2004, p. 94-100, ISBN 80-969202

[10] Zemčík P., Herout A., Crha L., Fučík O., Tupec P.: Particle rendering engine in DSP and FPGA, Proceedings of Engineering of Computer-Based Systems, Los Alamitos, US, IEEE CS, 2004, p. 423-428, ISBN 0-7695-2125-8

[11] Bryan L.: A Set of Definitions for Working with Spatial Filters, Proceedings of the 13th Student Conference and Competition EEICT, 2007 Volume 4, Brno, CZ, VUT Brno, 2007, p. 430-434, ISBN 80-214-3410-3

[12] Crha L., Fučík O., Zemčík P., Drábek V., Tupec P.: Inter chip communicating system with dynamically reconfigurable hardware support, Proceedings of the 6th IEEE DDECS Workshop, Poznan, Poland, 2003, p. 311-312, ISBN 83-7143-557-6

[13] Venard O., Blanchard Y., Lionti R., Crha L: Single chip FPGA realization of a 2D multicomp. wavelet transform, IEEE ISISPA, Rome, 2003, ISBN 953-184-062-8

[14] Crha L.: Jak se píše procesor, ABC Linuxu, Vol. 2005, Praha, CZ, ISSN 1214-1267, http://www.abclinuxu.cz/clanky/programovani/jak-se-pise-procesor

[15] Crha L: 2D Multicomponent Wavelet Transform, Preprints of IFAC Workshop PDS , FEI VŠB, Ostrava, CZ, 2003, p. 384-390, ISBN 0-08-044130-0

[16] Crha L: CPLD, FPGA and DSP communication, Proceedings of the 9th Student Conference and Competition EEICT, Brno, CZ, 2003, p. 619-623, ISBN 80-214-2379

Other publications

[17] Forsyth D., Ponce J.: Computer Vision: A Modern Approach, Prentice Hall, 2003, ISBN 978-0130851987

[18] Leibson S.: Challenges in Consumer Electronics for 21st Century, Keynote lecture, Worldcomp 2007, Las Vegas, NV, USA, June 2007

[19] Govindu G., Zhuo L., Choi S., Gundala P., Prasanna V.: Area and power performance analysis of a floating-point based application on FPGAs, Proceedings of the 7th Annual Workshop on High Performance Embedded Computing, 2003

[20] Jahne B., Hausecker H., Geisler P.: Handbook of Computer Vision and Applications, Academic Press, San Diego 1999, ISBN 0-12-379770-5

[21] Porter R.: Evolution on FPGAs for Feature Extraction, PhD Thesis, Queensland University of Technology, Brisbane, Australia, 2001

[22] Duda R., Hart P., Stork D.: Pattern Classification, Second edition, John Wiley & Sons Inc., New York NY, 2000, ISBN 0-471-05669-3

[23] Mariano V., Min J., Park J., Kasturi R., Mihalcik D., Li H., Doermann D., Drayer T.: Performance Evaluation of Object Detection Algorithms, Proceedings of the 16th ICPR, IEEE Computer Society, 2002, ISBN 1051-4651/02

[24] Qiang L., Bo Z.: Template Matching Based on Image Gray Value, Visual Communications and Image Processing, Proceedings of the SPIE, Volume 5960, 2005, p. 614-622, 2005SPIE.5960..614L

RESUME

Education

PhD degree at University of Technology, Brno, The Faculty of Informatics, started 2002, expected 2007

Ing. Degree (equivalent to Master's degree) at University of Technology, Brno, Czech Republic, The Faculty of *Electrical Engineering and Computer Science, 1997 – 2002*

- Diploma thesis at ESIEE Paris, France

Teaching experience

Penn State Erie, the Behrend College, Pennsylvania, 2005 – 2006

University of Technology Brno, Faculty of Electrical Engineering and Computer Science, 2002 – 2005

Industry experience

HW designer – 2002 – 2005, Camea Brno, www.camea.cz

- Traffic-related applications using cameras

English language

- Fluent in English
- One and half years of working experience in the US
- Passed General State Exam in English in the Czech Republic

ABSTRACT

This thesis presents a new hardware designed object detection method. The core of the proposed method is a highly parallel feature extraction performed by simple template matching elements. Due to the high parallelism the proposed method is suitable for hardware implementation. Part of the thesis is also an experimental architecture based on FPGA implementation. The proposed method can be enhanced by an adaptive learning system that automatically sets the method parameters according to environment conditions. An adaptive learning system implementation using FPGA dynamic reconfiguration is suggested in the thesis. To show that the proposed method can be used for real life situations, a case study dealing with license plate detection has been evaluated.