ATTRIBUTES SELECTION FOR LICENCE PLATE RECOGNITION BASED ON DECISION TREES

Aleš JANOTA, Jiří ZAHRADNÍK, Juraj SPALEK
Department of Control and Information Systems, Faculty of Electrical Engineering, University of Žilina,
Univerzitná 8215/1, 010 26 Žilina, Slovak Republic, tel. 041/513 3300,
E-mail: {ales.janota | jiri.zahradnik | juraj.spalek}@fel.utc.sk

SUMMARY

The paper deals with an image-based recognition system, which captures, interprets, records, and processes the image of a license plate for use in a variety of ITS applications. Such a system can save money by collecting and processing vehicle data without human intervention. Motorists are then allowed to pass toll plazas or weigh stations without stopping, which can save their time and prevent occurrence of congestions. Helping control access to secured areas or assisting in law enforcement can also improve safety and security. In different references this technology is also referred as Automatic Vehicle Identification, Car Plate Recognition, Automatic Number Plate Recognition, Car Plate Reader or Optical Character Recognition for Cars.

The paper discusses a concept of the Licence-Plate Recognition system consisting of several modules that are in different stages of development. It focuses on processing of images with only one vehicle captured. The system is based on decision trees created using inductive tree algorithms. After brief outline of the system the main attention is paid to selection and detailed description of attributes used to build a decision tree. The paper includes an algorithm used to find holes and arcs in the characters. Totally 32 principal attribute types are explained. Some of them are used several time in an analogical way, thus the final decision tree of the discussed system prototype contains 77 nodes. The system has been prototyped using C++ and designed for recognition of Slovak-style license plates. Some of introductory procedures (image input and selection of area with licence plate in the image) are still performed manually and in future can be fully automated.

Keywords: *image processing, plate recognition, decision tree, attribute, intelligent transportation system*

1. INDRODUCTION

Most of applications used in Intelligent Transportation Systems need to identify vehicles first. License-Plate Recognition (LPR) is an image-processing technology used to identify vehicles by their license plates. In different references this technology is also referred as Automatic Vehicle Identification, Car Plate Recognition, Automatic Number Plate Recognition, Car Plate Reader or Optical Character Recognition for Cars. Vehicle's LPR system has been a special area of interest in video surveillance domain for more than a decade or so. It is gaining popularity in various security and traffic applications, e.g. in controlling access to a toll collection point or a parking lot or integrated to the video vehicle detection systems which usually are installed in places of interest for intersection control, traffic monitoring etc., to identify vehicles that violate traffic laws or to find stolen vehicles [1]. There are a number of techniques used so far for recognition of number plates such as BAM (Bi-directional Associative Memories) neural network character recognition [2][3], pattern matching [4] etc. Most LPR systems focus on the processing of images with only one vehicle [5][6][7][8], others are able to process more vehicles at once [9]. The paper discusses a concept of the LPR system consisting of several modules that are in different stages of development. It focuses on processing of images with only one vehicle having Slovak-style license plate. The system is based on decision trees created using inductive tree algorithms [13].

2. CONCEPT OF THE LPR SYSTEM

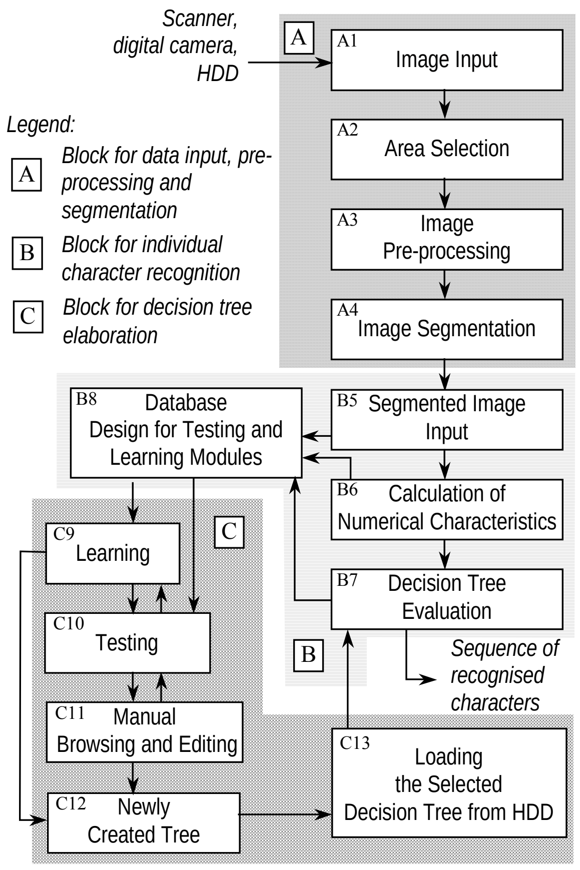
Scanner, digital camera, HDD
A
A1 Image Input
A2 Area Selection
A3 Image Pre-processing
A4 Image Segmentation
Legend:
A Block for data input, pre-processing and segmentation
B Block for individual character recognition
C Block for decision tree elaboration
B8 Database Design for Testing and Learning Modules
B5 Segmented Image Input
B6 Calculation of Numerical Characteristics
B7 Decision Tree Evaluation
B
Sequence of recognised characters
C
C9 Learning
C10 Testing
C11 Manual Browsing and Editing
C12 Newly Created Tree
C13 Loading the Selected Decision Tree from HDD

Fig. 1 Block scheme of the LPR system

To explain a concept of the proposed LPR system we can use the block scheme given in Fig. 1. The system consists of the blocks A, B and C containing several sub-blocks each.

In the sub-block *A*1 an image may be taken from a digital camera or a scanner connected via TWAIN interface [10] or loaded from the HDD as a 24-bit colour BMP file. The quality of the image underpins the entire system's accuracy and precision. To copy the image block-by-block the function `memcpy` from the library `<string.h>` is used [11]. Obviously, a machine can only identify a license plate's alphanumeric content after it has properly recognized that a plate is present in the field of view. Therefore, the main task of the sub-block *A*2 is to find an area containing single-line text containing a number of the license plate somewhere in the image.

The area selected in the previous step is pre-processed in the sub-block *A*3 using a fixed threshold and consequently copied into a new smaller image.

The sub-block *A*4 is used to find individual characters in a newly defined image. The algorithm starts in the centre of the vertical projection histogram and traverses up and down; in both directions significant change (decrease) of black colour is to be identified (Fig. 2). If difference between two adjacent points of vertical projected histogram is greater than 20 % of the width of the searched area, this point is chosen as a starting point (if going up) or ending point (if going down). Intensity of searched decrease is an experimentally found value. The algorithm works reliably with the text line not being inclined and characters being on the same level all.

Consequently the horizontal projection histogram of the found area can be calculated. The algorithm continues traversing from the left to the right side searching for the first non-zero value of the horizontal projection diagram. This value is to be marked as a starting point of the character. Other non-zero values are considered to be a part of the character. Reaching another zero value the area between two zero values is added to the list of segments. The algorithm adds another segment until the border of the area is reached.

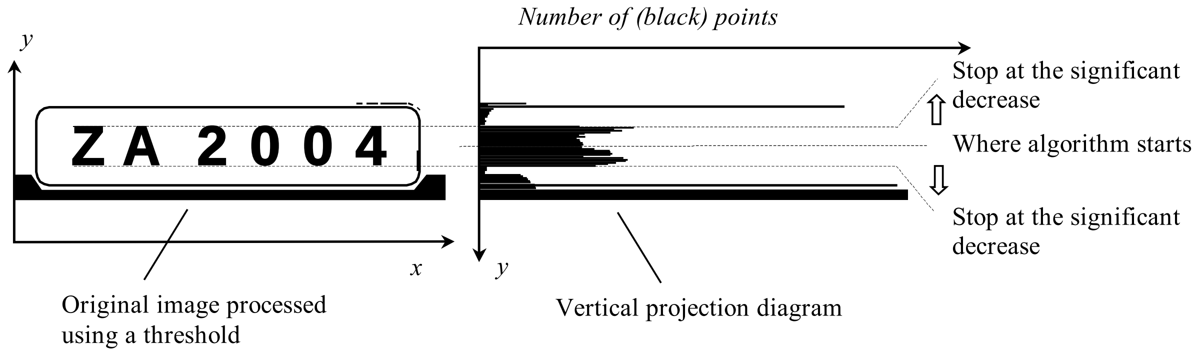
Number of (black) points
y
ZA 2004
x
y
Stop at the significant decrease
Where algorithm starts
Stop at the significant decrease
Original image processed using a threshold
Vertical projection diagram

Fig. 2 Searching for upper and lower text boundary

Thereafter those areas that contain too few image points necessary for inclusion of a character will be removed from the list of segments. The main disadvantage of the algorithm is its inability to separate coupled characters. A similar segmentation algorithm is presented in [12].

Fig. 3 Example of found segments

After segmentation (in the sub-block *B*5) the following data is delivered to the system:
- A pointer to the original image;
- A pointer to the list of found symbols (segments).

The sub-block delivers all found segments one after another. The structure of found segments is as follows:

```
struct area{
   int x1; //left upper corner
   int y1; //
   int x2; //right lower corner
   int y2; //
};

//structure of the list of found
segments
struct area_1{
   area *rn_i;    //one segment
   area_1 *next;  //pointer to the next
segment
};
```

For each identified segment, containing just one symbol, different numerical characteristics (attributes) may be calculated. An item of the attribute vector consists of the name, value and initiation. The structure of the attribute vector is as follows:

```
Typedef struct variable
{  chat   name[255];   //name
   int    value;       //value
   int    ini;         //initiation
   variable *n_var;    //pointer to the
next item };
```

The sub-block *B*6 is designed to meet the following requirements:

1. Provide simple interface for work with an attribute vector:
    - Setting up a new attribute vector;
    - Adding a new attribute into the attribute vector;
    - Removing an attribute from the attribute vector;
    - Finding an attribute contained in the attribute vector;
    - Deleting the whole attribute vector;
    - Writing items of the attribute vector on HDD;
2. Provide functions for calculation of attributes.

Maximum number of attributes contained in the attribute vector is unlimited.

Then a decision tree can be evaluated. Correct operation of the sub-block *B7* depends on fulfilling two conditions:

- Decision tree must be loaded from HDD;
- A calculated attribute vector must be available.

The sub-block provides the following functions:

- It makes reading of the decision tree from HDD possible;
- It makes classification of the image of input symbol possible based on the use of decision tree and attribute vector.

Functions relevant to this module are implemented in the file `read_tree.cpp`. Function used to load decision tree to memory is:

```
int link_fulltree(char *dir,char
*prj_file,NODE **root);
```

where:
`dir` – path to directory containing decision tree;
`prj_file` – path to the file of decision tree project;
`root` – function will return root node of loaded tree.

Function used to unload decision tree from memory s:

```
void rm_fulltree(NODE *root);
```

where:
`root` – pointer to the root of unloaded tree.

Function used to traverse decision tree is:

```
int a_tree(NODE *root,var
*var_list,NODE **decision)
```

where:
`decision` – function will return pointer to the leaf node;
`root` – pointer to root node of the tree;
`var_list` – pointer to the attribute vector.

The last sub-block *B8* of the "B" module enables design of a database for testing and learning.

The "C" module is designed to perform elaboration of a decision tree, i.e. to make procedures of learning, testing, manual browsing and editing of the newly created decision tree possible.

3. ATTRIBUTES USED TO CREATE A DECISION TREE

To find holes and arcs in the characters we use the following algorithm that consists of the following steps:

1. Memory allocation for four fields having dimensions 6×11.
2. Transiting a segment in order to get coordinates "to the left" and "to the top".
    - Take the first point with coordinates $B(x = 1, y = 1)$.
    - If $B(x, y) = 0$ then $L(x, y) = 0$, $U(x, y) = 0$.
    - If $B(x, y) = 1$ then
        - If $x = 1$ then $L(x, y) = 1$ else $L(x, y) = L(x\text{-}1, y) + 1$.
        - If $y = 1$ then $U(x, y) = 1$ else $U(x, y) = U(x, y\text{-}1) + 1$.
    - Transit all elements in the field $B(x, y)$.
3. Transiting the segment in order to get coordinates "to the right" and "to the bottom".
    - Take the first point with coordinates $B(x = 6, y = 1)$.
    - If $B(x, y) = 0$ then $R(x, y) = 0$, $D(x, y) = 0$.
    - If $B(x, y) = 1$ then
        - If $x = 6$ then $R(x, y) = 1$ else $R(x, y) = R(x+1, y) + 1$.
        - If $y = 11$ then $D(x, y) = 1$ else $D(x, y) = D(x, y+1) + 1$.
    - Transit all elements in the field $B(x, y)$;

where:

$B(x, y)$	- binary image function of the segment $x = 1, 2, \ldots,6; y = 1, 2, ..., 11$;
$B(x, y) = 0$	- the point with coordinates x, y has a colour of the character (black);
$B(x, y) = 1$	- the point with coordinates x, y has a colour of the background (white);
$L(x, y)$	- field for left direction;
$R(x, y)$	- field for right direction;
$U(x, y)$	- field for up direction;
$D(x, y)$	- field for down direction.

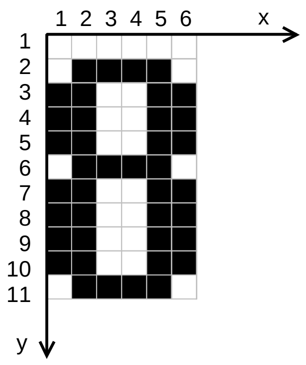

Fig. 4 Example of a segmented image

The use of this algorithm can be presented in application to the image of 6x11 points containing image of the number 8 (see Fig. 4).

[LEFT]	[RIGHT]	[UP]	[DOWN]
1,2,3,4,5,6,	6,5,4,3,2,1,	1,1,1,1,1,1,	2,1,1,1,1,2,
1,0,0,0,0,1,	1,0,0,0,0,1,	2,0,0,0,0,2,	1,0,0,0,0,1,
0,0,1,2,0,0,	0,0,2,1,0,0,	0,0,1,1,0,0,	0,0,3,3,0,0,
0,0,1,2,0,0,	0,0,2,1,0,0,	0,0,2,2,0,0,	0,0,2,2,0,0,
0,0,1,2,0,0,	0,0,2,1,0,0,	0,0,3,3,0,0,	0,0,1,1,0,0,
1,0,0,0,0,1,	1,0,0,0,0,1,	1,0,0,0,0,1,	1,0,0,0,0,1,
0,0,1,2,0,0,	0,0,2,1,0,0,	0,0,1,1,0,0,	0,0,4,4,0,0,
0,0,1,2,0,0,	0,0,2,1,0,0,	0,0,2,2,0,0,	0,0,3,3,0,0,
0,0,1,2,0,0,	0,0,2,1,0,0,	0,0,3,3,0,0,	0,0,2,2,0,0,
0,0,1,2,0,0,	0,0,2,1,0,0,	0,0,4,4,0,0,	0,0,1,1,0,0,
1,0,0,0,0,1,	1,0,0,0,0,1,	1,0,0,0,0,1,	1,0,0,0,0,1,

Fig. 5 Fields of distances

The algorithm will result in 4 fields (see Fig. 5) having a size of the original image, each field for one direction of transition. Item values in each field represent distances to the end of segment or distance to the edge for direction corresponding to a given point. Points belonging to the character image (i.e. black points) have item values equal to 0.

After algorithm application the fields $L(x, y)$, $R(x, y)$, $U(x, y)$, $D(x, y)$ are available in the memory. They are consequently used to calculate attributes:

1. Ratio: number of points bounded from all the sides – to – total number of segment points;
2. Ratio: number of points unbounded from the left – to – total number of segment points;
3. Ratio: number of points unbounded from the right – to – total number of segment points;
4. Ratio: number of points unbounded from the top – to – total number of segment points;
5. Ratio: number of points unbounded from the bottom – to – total number of segment points.

Given attributes are added to the attribute vector. They are applied separately for upper and lower half of the segment. In addition other attributes are added:

6. Ratio: number of points bounded from all sides in the upper (lower) half of the segment – to – total number of segment points;
7. Ratio: number of points unbounded from the left in the upper (lower) half of the segment – to – total number of segment points;
8. Ratio: number of points unbounded from the right in the upper (lower) half of the segment – to – total number of segment points;
9. Ratio: number of points bounded from the bottom in the upper (lower) half of the segment – to – total number of segment points;
10. Ratio: number of points unbounded from the top in the upper (lower) half of the segment – to – total number of segment points.

Another possible calculation is based on different shapes of character edges. Particular values of attributes are calculated separately for each edge vector of distances (distances of points being closest to the segment edge).

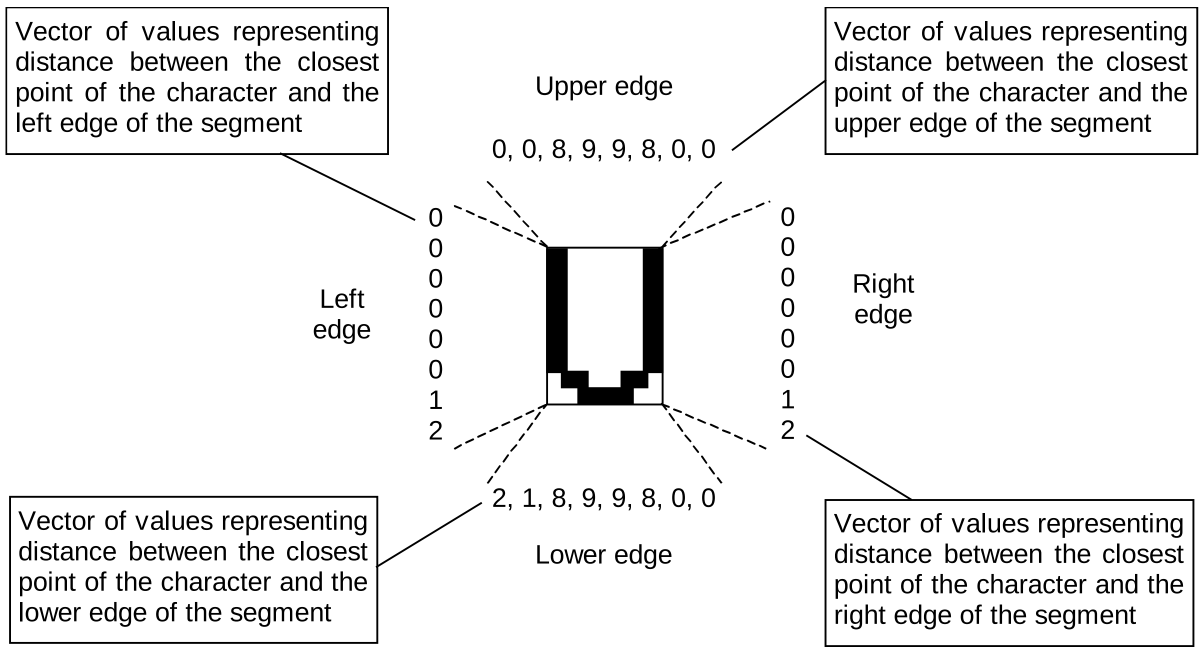
Vector of values representing distance between the closest point of the character and the left edge of the segment
Upper edge
0, 0, 8, 9, 9, 8, 0, 0
Vector of values representing distance between the closest point of the character and the upper edge of the segment
0 0 0 0 0 0 1 2
Left edge
0 0 0 0 0 0 1 2
Right edge
2, 1, 8, 9, 9, 8, 0, 0
Lower edge
Vector of values representing distance between the closest point of the character and the lower edge of the segment
Vector of values representing distance between the closest point of the character and the right edge of the segment

Fig. 6 Vectors of "closest point-symbol edge" distances

To capture edges of the character several attributes have been chosen, using the following symbols:

h	- height of the character in points
w	- width of the character in points
$L(i)$	- vector of the left side, where $i = 1, 2, \ldots, h$
$R(j)$	- vector of the right side, where $j = 1, 2, \ldots, h$
$U(k)$	- vector of the upper side, where $k = 1, 2, .., w$
$D(l)$	- vector of the lower side, where $l = 1, 2, .., w$

a) Edges directing to the left and to the right

For the left side we can write:

If $L(i) - L(i\text{-}1) < 0$ then the edge in the point i leads to the left (e.g. Fig. 7a);

If $L(i) - L(i\text{-}1) > 0$ then the edge in the point i leads to the right (e.g. Fig. 7b);

If $L(i) - L(i\text{-}1) = 0$ then the edge in the point i doesn't change its direction (e.g. Fig. 7c).

Transiting all the points of the relevant vector the following attributes are added to the attribute vector:

11. Ratio: number of points directing for the left – to – length of the edge;
12. Ratio: number of points directing for the right – to – length of the edge;
13. Ratio: number of points that do not change their direction – to – length of the edge.

The same approach can be applied for the right side.

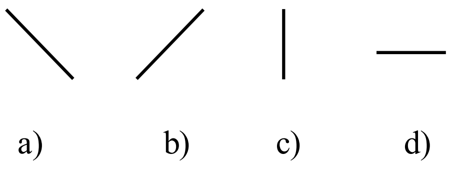

Fig. 7 Possible direction of edges

b) Edges directing up and down

For the lower side we can write:

If $D(l) - D(l\text{-}1) < 0$ then the edge in the point i leads down (e.g. Fig. 7a);

If $D(l) - D(l\text{-}1) > 0$ then the edge in the point i leads up (e.g. Fig. 7b);

If $D(l) - D(l\text{-}1) = 0$ then the edge in the point i doesn't change its direction (e.g. Fig. 7d).

Transiting all the points of the relevant vector the following attributes are added to the attribute vector:

14. Ratio: number of points directing up – to – length of the edge;
15. Ratio: number of points directing down – to – length of the edge;
16. Ratio: number of points that do not change their direction – to – length of the edge.

The same approach is applied for the upper side.

These attributes cannot be used to describe exactly shapes of edges but can enable making decision whether edge directing is monotonous or not (e.g. characters **A, U**, **V** for the left and right side). In the case of a monotonous nature the attributes may be used to quantify how sudden fall of the edge is and thus distinguish between characters, e.g. **U** and **V**. However, these attributes can be applied after assumption of shape and monotony expressed in the decision tree.

c) Notation of a shape of the edge using vector of values representing downward-sloping and upward-sloping (analogically for direction of the edge to the left and to the right)

Shape of the edge is expressed for every edge (left, upper, right, lower) separately using eight attributes. For lower side:

17. Upward-sloping 1 (*ds_u*1)
18. Downward-sloping 1 (*ds_d*1)
19. Upward-sloping 2 (*ds_u*2)
20. Downward-sloping 2 (*ds_d*2)
21. Upward-sloping 3 (*ds_u*3)
22. Downward-sloping 3 (*ds_d*3)
23. Upward-sloping 4 (*ds_u*4)
24. Downward-sloping 4 (*ds_d*4)

Attributes represent ratio of downward (upward) sloping intensity or points with no change of direction to length of the edge. If the edge contains more than four changes of direction the sum of percentage of downward sloping and upward sloping less than 100.

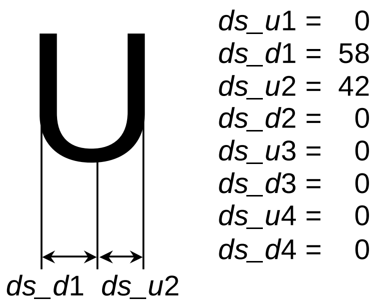

Fig. 8 Expressing shape of the edge

From Fig. 8 it is apparent that the lower edge begins downward sloping that covers 58 % of its length, then the edge begins upward sloping covering 42 %.

d) Maximums of histograms of projection profiles

Maximums and minimums of histograms are calculated for horizontal and vertical projection profiles and separately also for each third of the profile (Fig. 9). They can be used to identify long vertical and horizontal lines. The main limitation of this attribute consists in the fact that any picture slewing or sloping change perpendicular character of lines.

Calculated maximus and minimus are used to create another items of an attribute vector:

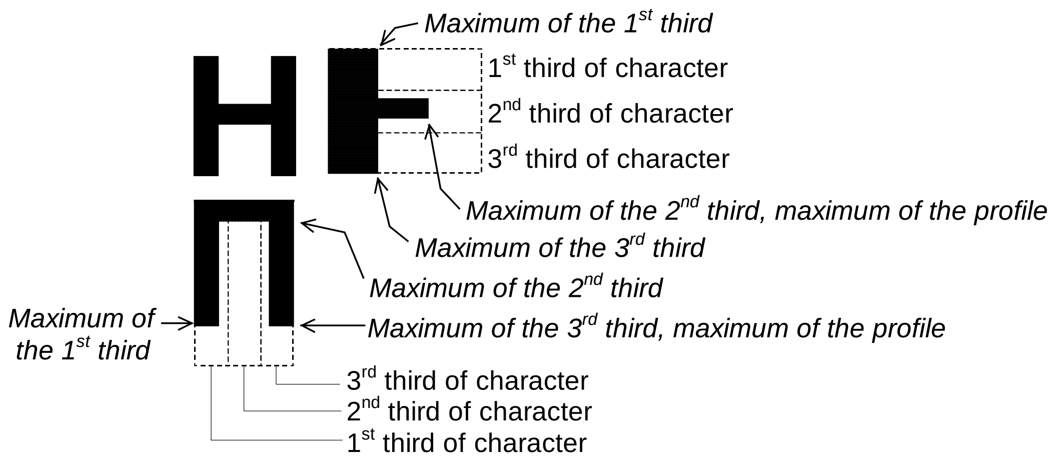

Fig. 9 Maximums of histograms of projection profiles

25. Ratio: maximum of horizontal projection histogram – to – width of the character;
26. Ratio: maximum of horizontal projection histogram in the 1st third of the character – to – width of the character;
27. Ratio: maximum of horizontal projection histogram in the 2nd third of the character – to – width of the character;
28. Ratio: maximum of horizontal projection histogram in the 3rd third of the character – to – width of the character;
29. Ratio: maximum of vertical projection histogram – to – height of the character;
30. Ratio: maximum of vertical projection histogram in the 1st third of the character – to – height of the character;
31. Ratio: maximum of vertical projection histogram in the 2nd third of the character – to – height of the character;
32. Ratio: maximum of vertical projection histogram in the 3rd third of the character – to – height of the character.

4. CONCLUSION

Authors presented the concept of a plate recognition system and described the actual state of its development. Special attention was paid to selection and description of attributes used to create a proper decision tree. The first version of the decision tree was constructed using Quinlan's ID3 algorithm followed by some other manual modifications (pruning). Actual version of the tree consists of 77 nodes.

This work has partially been supported by the Grant Agency of the Slovak Republic VEGA, grant No. 1/1044/04 "*Theoretical Foundations for Implementing e-Safety Principles into Intelligent Transportation Systems*" and partially by the institutional research project No. 07/604/2005 "*Transport telematics and tools to improve its quality*" solved at the Faculty of Electrical Engineering, University of Žilina.

REFERENCES

[1] Xin Li, XiaoCao Yao, Yi L. Murphey, Robert Karlsen, Grant Gerhart: A Real-Time Vehicle Detection and Tracking System in Outdoor Traffic Scenes, *Proc. of Pattern Recognition, 17th International Conference on ICPR'04, Volume 2,* August 2004, pp. 761-764.

[2] Maged M. M. Fahmy: Automatic number-plate recognition: neural network approach, *Proc. of Vehicle Navigation and Information Systems Conference*, September 1994, pp. 99-101.

[3] S. Draghici: A Neural Network Based Artificial Vision System for Licence Plate Recognition, *International Journal of Neural Systems*, Vol. 8, No. 1, , 1997, pp. 113-126.

[4] D. Irecki & D. G. Bailey: Vehicle registration plate localization and recognition, *Proc. of the Electronics New Zealand Conference, ENZ Con '01*, New Plymouth, New Zealand, September 2001.

[5] X. F. Hermida, F. M. Rodríguez, and J. L. Fernandez: A System for the Automatic and Real Time Recognition of V.L.P.'s (Vehicle License Plates), *Proc. of International Conference on Image Analysis and Processing*, Florencia, Italia. September 1997.

[6] P. Comelli, et al.: Optical Recognition of Motor Vehicle License Plates, *IEEE Trans. on Vehicular Technology*, Vol. 44, No. 4, November 1995, pp. 790 –799.

[7] T. Naito, et al.: Robust License-Plate Recognition Method for Passing Vehicles under Outside Environment, *IEEE Trans. on Vehicular Technology*, Vol. 49, No. 6, November 2000, pp. 2309-2319.

[8] J. A. Hegt, R. J. de la Haye and N. A. Khan: A High Performance License Plate Recognition System, *Proc. of IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 5, October 1998, pp. 4357-4362.

[9] Hsi-Jian Lee, Si-Yuan Chen, Shen-Zheng Wang: Extraction and Recognition of License Plates of Motorcycles and Vehicles on

Highways, *Proc. of Pattern Recognition, 17th International Conference on (ICPR'04),* Vol. 4, August 2004, pp. 356-359.

[10] TWAIN Specification. Ver. 1.0, 2000, 552 p. http://www.twain.org/docs/Spec1_9_197.pdf.

[11] Herout, P.: Učebnica jazyka C. 2. diel, Kopp, České Budějovice, 1999 (in Czech).

[12] Kok Kiaw Teo: Low Cost Number Plate Recognition, University of Queensland, St. Lucia, Queensland, 2003.

[13] Dragúň, J.: Rozpoznávanie znakov s využitím techník umelej inteligencie. MSc. Thesis, Department of Control and Information Systems, FEE University of Žilina, 2004 (in Slovak).

BIOGRAPHY

Aleš Janota was born on 5. 4. 1963. In 1986 he graduated (MSc.) at the Department of Interlocking, Signalling and Communications of the Faculty of Mechanical and Electrical Engineering at the Technical University in Transport and Communications in Žilina. He defended his PhD. in the field of telecommunications in 1998; his thesis title was "Knowledge Base for Railway Interlocking and Signalling". In 2003 he was given the academic degree "Associate Professor" (doc.); his habilitation thesis was entitled "Formal specification and verification of safety-critical systems". At present he works as an associate professor at the Department of Control and Information Systems at the Faculty of Electrical Engineering of the University of Žilina and both his research and educational activities are focused mostly on AI techniques and their applications in the field of intelligent transportation systems; a special attention is also paid to theory of safety-related control systems.

Jiří Zahradník was born on 23. 7. 1946. In 1969 he graduated from the Department of Interlocking, Signalling and Communications at the Faculty of Mechanical and Electrical Engineering at the Technical University of Transport in Žilina. He defended his PhD thesis in the field of communication engineering in 1981. In 1987 he became an associate professor at the Faculty of Mechanical and Electrical Engineering of the Technical University of Transport and Communications in Žilina in the field of interlocking, signalling and communications in transport. In 2004 he habilitated; his habilitation thesis was entitled „Analysis of failure effects of computer safety systems". At present he works at the Department of Control and Information Systems at the University of Žilina and his research activities are aimed at technologies and services of intelligent transportation systems and the solving problems of their safety.

Juraj Spalek was born on 22. 6. 1953. In 1976 he received his MSc. from the Department of Interlocking, Signalling and Communications at the Faculty of Mechanical and Electrical Engineering of the Technical University of Transport and Communications in Žilina. His PhD thesis (1981) entitled "Electronic logical system as interlocking system" concerned the area of safety engineering. His research and pedagogical interests include reliability engineering, fuzzy-set theory applications, and analysis of reliability and safety of electronic systems for safety-related critical applications. It was on this topic that he worked out his second doctoral thesis in 1993 "Electronic systems properties in critical applications". At the present he works as an associate professor and a head of the Department of Control and Information Systems at the Faculty of Electrical Engineering of the University of Žilina.