Proceedings of the Twelfth IASTED International Conference
Intelligent Systems and Control (ISC 2009)
November 2 - 4, 2009 Cambridge, MA, USA

THE USAGE OF COMPUTER VISION FOR IDENTIFYING VEHICLES DURING THEIR PRODUCTION PROCESSES

Fernando Mello[†], Leizer Schnitman[†], J. A. M. Felippe de Souza[††]

[†] † *Dept of Electrical Engineering,* Universidade Federal da Bahia (UFBA), Salvador, Bahia – Brazil
[††] *Dept of Electromechanical Engineering,* Universidade Beira Interior - UBI - Covilhã – Portugal

ABSTRACT
The automotive industry, motivated by its necessity of processes and products automation, has been using pattern recognition and computer vision to develop system that are able to perform tasks previously dedicated to the human. This paper discusses a system developed to recognize the Vehicle Identification Number from a picture taken from this vehicle during its production process. Noise factor such as ambient lightening variation and image resolution are treated and each software step is presented.

KEY WORDS
Pattern recognition; Computer vision; Image processing; Artificial neural network; Automotive industry; Barcode

1. Introduction

Tracking the vehicle during its production process, including its assembly, quality control and shipment to dealer is crucial to guarantee the whole manufacturing process integrity. The tracking process allows:

1. Generation of information to feed the logistic systems, responsible for controlling parts consumption and supply chain management;
2. Tracking separately the production stages of each vehicle ordered to the plant;
3. Recording the vehicle construction history, attaching to its report the results for all quality and integrity tests performed with this vehicle during its assemblage.

This paper proposes the employment of pattern recognition and computer vision to create a system which is able to recognize the Vehicle Identification Number (VIN) from a picture taken of a vehicle during its production process. To perform this task, the system has to recognize in the image a label stuck to the vehicle. This label is actually a barcode with the VIN. Once the label recognition is done the system has to decipher the barcode, converting into the VIN.

The aim of this system is to eliminate the human operator for performing the vehicle identification. This will cut out some aspects such as: mental fatigue and tedium associated to this sort of operation as well as reducing labor costs inherent to this activity. Technologies such as RFID (Radio-Frequency Identification) could be used to perform the vehicle identification task, but costs associated to it still represent a practical barrier.

Creating such a system which is able to decipher the barcode information printed on the stamp that identifies a vehicle imposes some challenges. To mention a few examples:

- Variation in the relative position between the vehicle and the camera in charge of acquiring the image. This causes variation on the object size that the system aims to identify (the barcode) and therefore causes variation on the object resolution.
- Varied illumination conditions due to weather change, such as sunny or rainy days, and also due to the time the image is acquired, such as day or night
- Discrepancies in both the stamp position and orientation, often due to incorrect sticking in the glass (since it is a manual operation) or also due to geometric differences among all vehicle models produced in the plant.

Some authors have presented papers related to barcode identification, such as Xiangju *et al* [1], Ohbuchi [2] and Youssef [3], but using instead images captured on a different ambient and usually with better quality than the images used in the application of this paper.

2. VIN Recognition System

The system has to recognize the stamp stuck on the windshield and decode the lower barcode stamped on it, as shown by Figure 1.

Figure 1 – Stamp that contains the barcode

To accomplish its task, the system was designed as a combination of six different modules, and this is shown in Figure 2.

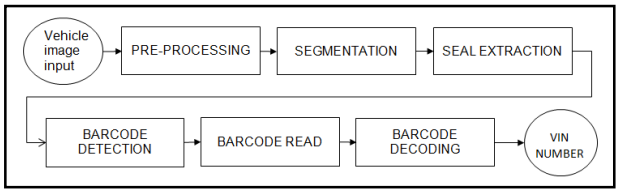

Figure 2 – System Modules

2.1 Pre-processing Module

This module has two main goals. The first one is to create other images, by reducing in 50% the original vehicle image and by converting the gray scale input image into a black and white image. The reduced resolution image generated by this module will be used to reduce computational processing effort from some modules that do not need to use the high definition images to accomplish their tasks. That allows reducing the overall cycle time for the system.

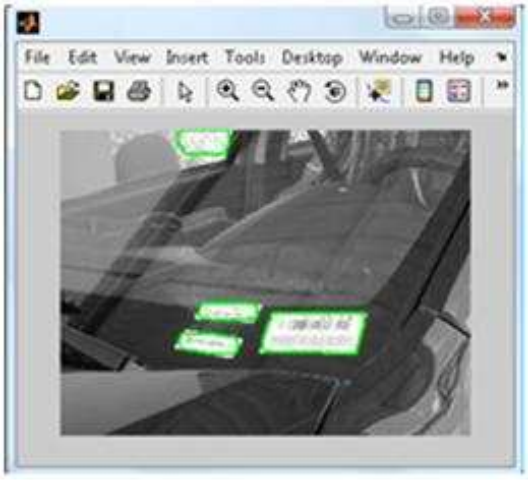

Figure 3 – Output image of segmentation module

The black and white image is used to support the second goal of this pre-processing module. The fact the stamp is printed in white paper gives to it the property of being one of the most intense objects in the picture. Using that characteristic, the module generates another image by filtering the black and white image using 80% of the maximum intensity level as a threshold. The image originated from this action contains a lot less objects than the original image and therefore it is used as an input for the next module, the segmentation module, reducing its processing effort (and therefore the processing time). Another important issue to notice is the employment of a percentage of the most intense pixel in the image and not a constant value to define the threshold value. This criterion gives robustness to support illumination variation as a noise factor.

2.2 Segmentation Module

The goal of this module is to identify on the vehicle image the areas that are prone to contain the stamp (called candidate areas). It basically utilizes morphological characteristics of the stamp to extract from the filtered black and white image received from the previous module the boundary of regions that may contain the stamp in it.

The first step taken by this module is to eliminate the areas (group of with pixels) that have a smaller number of pixels than the minimum number expected for a stamp. After that, the module remove vertical sections composed by white pixels that do not sum the minimum number of pixels expected for a stamp vertical section. This process is repeated for the horizontal sections. After these three steps, all remaining areas are labeled and have their boundaries outlined in such a way they can be used by the stamp extraction module that has to define which one of this areas contains the stamp (Figure 3).

2.3 Stamp Extraction Module

As already mentioned in the previous section, the goal of this module is to elect the region in the vehicle image that contains the barcode stamp. In order to do that, this module employs a barcode characteristic that in fact it is composed by a large number of transitions between dark and bright pixels, belonging to the black bars and white spaces of the code.

The module applies a calculation that punctuates each pixel from the candidate areas according to the number of transitions [4] between dark and clear pixels it has on its neighborhood; equation (1).

$$S_i = \sum_{n=n_{min}}^{n_{max}} a * |p_n - p_{n-1}| * e^{-b*d_{in}} \qquad (1)$$

where, S_i is called "transition density" and it is a assigned value to a pixel i from the candidate area being analyzed. n_{min} is the column of the further pixel from the left border of the candidate area. n_{max} is the column of the further pixel from the right border of the candidate area. p_n is the intensity value of a pixel n from the same line as pixel i. d_{in} is the distance between pixel i and n, calculated by subtracting the column values of both pixels. a and b being positive constants.

To define among the candidate areas which one is the area that contains the stamp, the module applies a second step calculation. It sums the transition density of all pixels from the same candidate area and assigns this value to that area equation (2). The candidate area that presents the highest sum of transition density will be elected as the area that contains the barcode stamp.

$$T = \sum_{m=m_{min}}^{m_{max}} \sum_{n=n_{min}}^{n_{max}} S_{m,n} \qquad (2)$$

where T is the sum of transition density of all pixels from the candidate area being analyzed. m_{min} is the row value of the highest pixel from the top border of the candidate area. m_{max} is the row value of the lowest pixel from the bottom border of the candidate area. n_{min} is the column of the further pixel from the left border of the candidate area. n_{max} is the column of the further pixel from the right border of the candidate area. $S_{m,n}$ is the transition density given to pixel positioned at row m and column n.

The usage of this process as a classifier for candidate areas presents very robust results. In all 34 images tested the area that contains the stamp presented a T value about

10 times bigger than the other candidate area. Figure 4 shows the T value for each one of the candidate areas from a vehicle image.

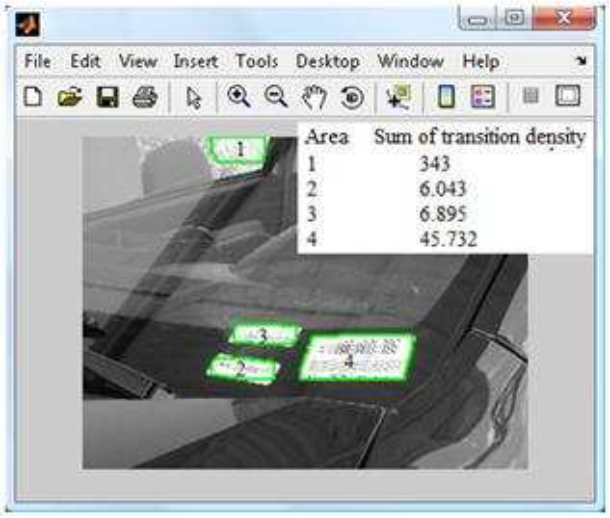

Figure 4 – Sum of transition density for each candidate area

2.4 Barcode Detection Module

Having the stamp area been detected, the next necessary phase is to define where the barcode is located within this area. To do that, the barcode detection module searches for bars from the barcode. It utilizes the transition density values calculated in the previous module and define the pixel that has the highest value. Since the barcode pixels have higher density transition values than any other pixels, the module starts the search for bars from the pixel presenting the highest density transition value. That is because this pixel is in a region that tends to have a large number of bars in its neighborhood.

This module moves left and right from the start pixel searching a bar border. It uses the black and white image and simply compares the pixel value with its right neighbor. If the analyzed pixel is equal to zero (is black) and its right neighbor is one (is white), that means the software found a bar border. This bar's boundary is traced and the software starts searching for a new bar (Figure 5). It stops after mapping five bars, which will be used by the next module for reading the barcode.

2.5 Barcode Read Module

This module aims to read and store the intensity value of all pixels located under a constructed line that crosses transversally all bars and spaces that constitute the barcode. The first step to accomplish this objective is constructing the transversal line. For doing that the module defines the lowest points of all five bars located by the barcode detection module. Using the minimum square method [5], the software uses the row and column coordinates of each point, called x_i and y_i respectively in equation (3), generating a line that theoretically contains all these five points.

$$\begin{cases} \left(\sum_{i=1}^{n} x_i\right) a + \left(\sum_{i=1}^{n} x_i^2\right) b = \sum_{i=1}^{n} x_i y_i \\ na + \left(\sum_{i=1}^{n} x_i\right) b = \sum_{i=1}^{n} y_i \end{cases} \quad (3)$$

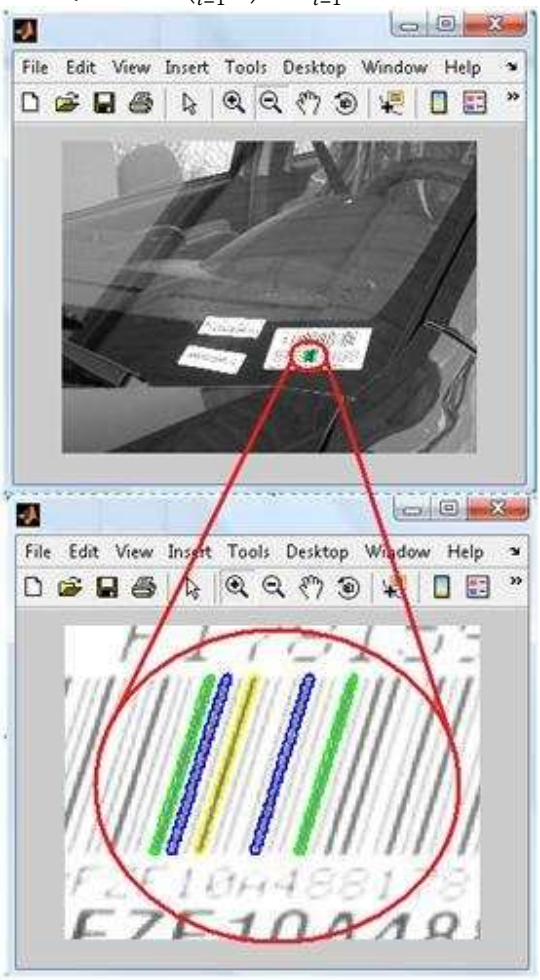

Figure 5 – Output image of barcode detection module.

Figure 6 – Two lines transversal to de barcode.

In this system: a is constant and represents the line slope whereas b represents the independent term of the

line equation. Since these five points are not perfectly collinear due to stamp printing process, image quality, image acquiring condition, etc., the line traced may present a slope that makes this line not crossing the bars that are further from the barcode center. In order to avoid that the software traces a new line, parallel to the first one but passing over the center of the first bar detected by the previous module (Figure 6).

This second line is used as a path to read the barcode pixels. Starting from the pixel at the center of the first bar detected, the column value (here treated as y axel of a Cartesian plane) is unitarily incremented and the row value (here treated as x axel of the Cartesian plane) calculated using the line equation. Having the address of a pixel under the line been defined, this pixel intensity is stored. This operation is repeated up to the right end of the barcode. After that, the same is done for the left portion of the barcode, in such a way that all pixels that belong to bars and spaces of the barcode and are positioned under the transversal line had their intensity read and stored.

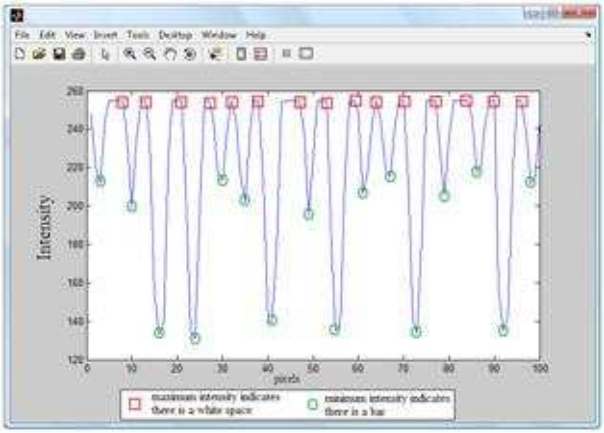

Figure 7 – Pixel surrounded by square belongs to white space, by circle belongs to bar.

2.6 Barcode Decoding Module

This module has to classify each pixels read by the previous module in a pixels that belongs to a bar or to a space from the barcode. After that, it is necessary to classify each one of the bars in large width or thin width and classify each one the spaces in a large or thin space. Having all that done, the module groups a sequence of five bars interspersed by four white spaces and convert this group into the correspondent character.

The previous module used a gray scale picture for reading the barcode pixels intensity. Therefore, the values stored may vary from 0 to 255. It is necessary to establish a method for defining which pixels belong to bars and which pixels belong to spaces. Using a fixed value as a threshold value limits the software robustness to lightening condition. The solution adopted uses a percentage of maximum intensity of the closest white space to classify the pixels.

Does not matter under what lightening condition an image is acquired, the bars from the barcode will always be darker than the spaces. This software uses this property to define all white spaces from the barcode. It goes through the vector that stores the barcode pixels intensity searching for local maximum values. Local maximum values indicate pixels which intensity is higher than the pixels at its right and left side. These pixels that present a local maximum intensity are pixels that belong to bars, no matter what intensity they have. Figure 7 shows a plot of intensity values from the pixels stored in the vector originated by the barcode read module. Pixels that represent a local maximum and local minimum are identified. They belong to white spaces and bars respectively.

For classifying all pixels as pertaining to a bar or to a white space, the software compares the intensity of this pixel with the intensity of the local maximum intensity pixel closest to it. If the analyzed pixel presents up to 98% of the local maximum intensity pixel, it is considered as a pixels that belongs to a bar. If not, it is considered a pixel that belongs to a space. This process allows the software classifying all barcode pixels.

The next step is classifying all bars in large or thin bars and all white spaces in thin or large white spaces. The bars classifier is a feedforward backpropagation Artificial Neural Network (ANN) [6]. It has eight neurons in the hidden layer and one in the output layer. Neurons from hidden layer utilize hyperbolic tangent sigmoid transfer function while the output layer's neuron uses a linear transfer function. This Neural Network receives two variables as input. The first and second input are:

1. Average intensity of all pixels that compose a certain bar, divided by the average intensity of four known large bars; equations (4) and (5);
2. Number of pixels that compose a certain bar, divided by the number of pixels that compose the whole barcode; equation (6).

$$E1_n = \frac{\frac{\sum_{p_n=1}^{j_n} i_{p_n}}{j_n}}{\bar{\imath}_{bc}} \tag{4}$$

$$\bar{\imath}_{bc} = \frac{\frac{\sum_{p_3=1}^{j_3} i_{p_3}}{j_3} + \frac{\sum_{4=1}^{j_4} i_{p_4}}{j_4}}{4} + \frac{\frac{\sum_{p_{93}=1}^{j_{93}} i_{p_{93}}}{j_{93}} + \frac{\sum_{p_{94}=1}^{j_{94}} i_{p_{94}}}{j_{94}}}{4} \tag{5}$$

where: $E1_n$ is the ANN input for a certain bar n. p_n is a certain pixels that constitute the bar n, i_{pn} is the intensity of pixel p_n and j_n is the number of pixels that compose the bar n; for equation (4). Also: the average intensity of known large bars number 3, 4, 93 and 94 is given by $\bar{\imath}_{bc}$; for equation (5).

Bars 3, 4 , 93 and 94 are previously known as large bars due to the fact Vehicle Identification Numbers start and finish with the asterisk character. Barcode standard utilized by automotive industry is called Code 39 [7]. In Code 39, the asterisk is composed by a sequence of five

bars, where the third and fourth bars are large and bars one, two and five are thin. Since the VIN is made of 19 digits and the first and the last are asterisk, bars number 3, 4, 93 and 94 are large bars. The operation of dividing the intensity of each bar by the average intensity of some large bars from the same image has the power of normalizing the results, enabling the usage of the ANN for any image, no matter what lightening condition the image was taken.

$$E2_n = \frac{np_n}{np_c} * 1000 \qquad (6)$$

where $E2_n$ is the second input variable to the ANN. np_n is the number of pixels that compose the bar n and np_c is the number of pixels that compose the whole barcode from where the bar n was extracted. Dividing the number of pixels that compose a certains bar by the total number of pixels from the barcode normalizes the result and allows the ANN to deal with images that were taken from different distances between the camera and the vehicle.

Figure 8 – Identified VIN.

As mentioned before, the Code 39 barcode represents each character by a sequence of five bars interspersed by four white spaces. The four white spaces are a combination of three thin white spaces plus one large white space. This characteristic allows the software to simply compare the number of pixels that compose each one of the four spaces from a character set in order to define which one is the large white space. That is basically the way this software classifies the white spaces in large or thin spaces. But due to the poor image conditions of some vehicle pictures, there are occurrences of large white spaces and thin white spaces from the same character set that present the same number of pixels. For dealing with this situation it was created a extra decision criterion, to be used only when this sort of uncertainness occurs. This criterion uses the percentage of intensity transition between the pixels from the white space and the left and right bars that surround the space; equation (7).

$$V_{ti} = \frac{\left(1-\frac{i_{pbbe}}{i_{pbee}}\right)+\left(1-\frac{i_{pbbd}}{i_{pbde}}\right)}{2} \text{ x } 100\% \qquad (7)$$

where: *Vti* is the intensity transition percentage, i_{pbbe} is the intensity of the border pixel from left bar, i_{pbee} is the intensity of the left border pixel from the white space, i_{pbbd} is the intensity of the border pixel from right bar, i_{pbde} is the intensity of the right border pixel from the white space. When comparing white spaces that present the same number of pixels, the software elects as the large white space the one with smaller *Vti*.

When all bars and spaces are classified, the software packages groups of five bars and four spaces, which correspond to a character set, and search in a nine dimensions matrix the character represented by that group. This nine dimensions matrix, where each dimension represents a bar or a space, encloses all characters that may exist in a VIN, positioned in their respective combination of bars and spaces ruled by the Code 39 barcode.

After repeating this process 19 times, the result is the VIN translated from the barcode format to our alphabetical characters, as shown by Figure 8.

3. Conclusion

The proposed system created to decode the Vehicle Identification Number from a picture taken from the vehicle during its production process successfully utilizes pattern recognition and computer vision concepts to perform its task. The system is able to deal with noise factors such as variation between the camera and the photographed vehicle, ambient lightening conditions and stamp positioning and orientation.

The success rate in decoding a character is deeply linked to the input image quality. It reaches 95.4% for images taken under the original setup conditions and rises up to 100% for improved images (as it is illustrated in Figures 10 and 11). Tests done showed the success rate is sufficiently good to support a commercial application development.

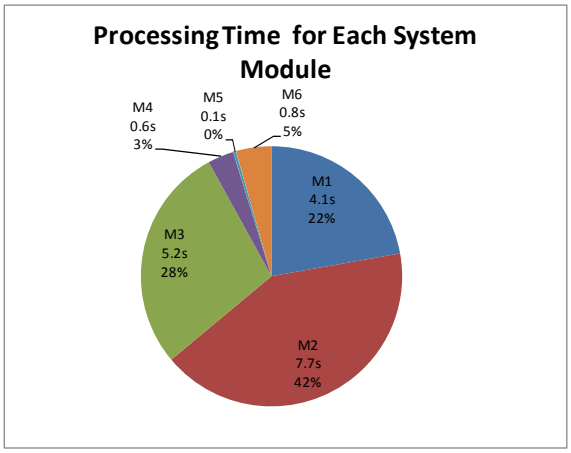

Figure 9 – Recognition results

The processing time is about 19 seconds per image (Figure 9) and good opportunities for reduction are available, mainly in segmentation and stamp extraction modules. The software is developed in MatLab, and the simple conversion from this ambient to C or C++ should respond with a considerable increase in processing time efficiency.

References

[1] L. Xiangju *et al*, *A robust barcode reading method based on image analysis of a hierarchical feature classification*, IEEE International Conference on Intelligent Robots and Systems, 2006.

[2] E. Ohbuchi *et al*, *Barcode readers using camera device in mobile phones*, International Conference on Cyberworlds, 2004.

[3] S. Youssef *et al*, *Automated barcode recognition for smart identification and inspection automation* Alexandria, Arab Academy for Science and Technology, 2006.

[4] S. L. Chang et al, *Automatic license plate recognition, IEEE Transactions on Intelligent Transportation Systems*, vol. 5, nr. 1, 2004.

[5] M. J. Souza, *Ajuste de curvas pelo método dos quadrados mínimos*. Ouro Preto, UFOP, Brazil, 2008.

[6] J. R. Pacheco, *Reconhecimento de padrões de vibração em máquinas rotativas utilizando rede neural artificial*. Salvador, UFBA, Brazil, 2007.

[7] J. R. Loeffler (Ford Motor Co). *Barcode standards aid Detroit*, American Machinist & Automated Manufacturing, v 130, n 12, Dec, 1986, p 73-74.

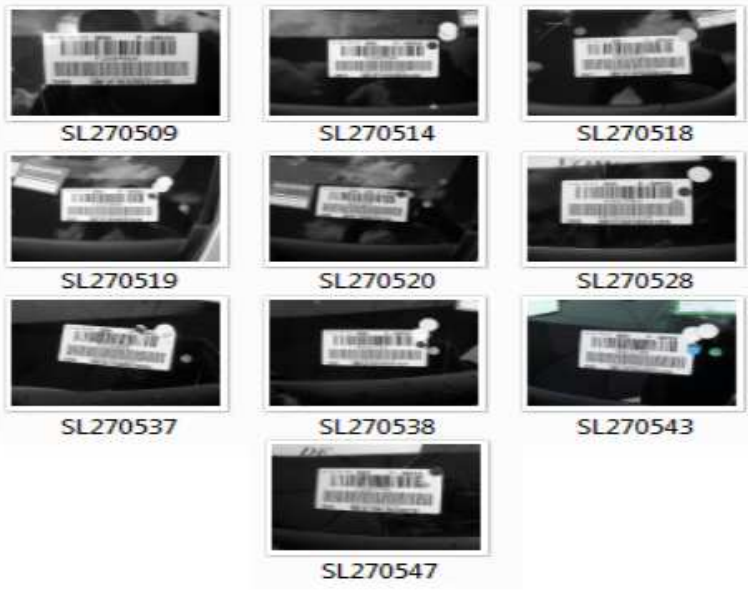

Figure 10 – Improved images.

Image ID	Reconized VIN	Number of unidentified characters	% of correct character recognition
SL270509.jpg	*9BFZF10A288260486*	0	100.0%
SL270514.jpg	*9BFZ10A488260358*	0	100.0%
SL270518.jpg	*9BFZE16P888949303*	0	100.0%
SL270519.jpg	*9BFZF10A488253958*	0	100.0%
SL270520.jpg	*9BFBT10N288252885*	0	100.0%
SL270528.jpg	*9BFZF20A188241370*	0	100.0%
SL270537.jpg	*9BFZF10A888270200*	0	100.0%
SL270538.jpg	*9BFZF20A388241371*	0	100.0%
SL270543.jpg	*9BFZF20A688241378*	0	100.0%
SL270547.jpg	*9BFZF10A188248779*	0	100.0%
		Average	100.0%

Figure 11 – Recognition results