Computers, Materials & Continua CMC, vol.64, no.1, pp.233-251, 2020

Classification for Glass Bottles Based on Improved Selective Search Algorithm

Shuqiang Guo[1, *], Baohai Yue[1], Manyang Gao[2], Xinxin Zhou[1] and Bo Wang[3]

Abstract: The recycling of glass bottles can reduce the consumption of resources and contribute to environmental protection. At present, the classification of recycled glass bottles is difficult due to the many differences in specifications and models. This paper proposes a classification algorithm for glass bottles that is divided into two stages, namely the extraction of candidate regions and the classification of classifiers. In the candidate region extraction stage, aiming at the problem of the large time overhead caused by the use of the SIFT (scale-invariant feature transform) descriptor in SS (selective search), an improved feature of HLSN (Haar-like based on SPP-Net) is proposed. An integral graph is introduced to accelerate the process of forming an HBSN vector, which overcomes the problem of repeated texture feature calculation in overlapping regions by SS. In the classification stage, the improved SS algorithm is used to extract target regions. The target regions are merged using a non-maximum suppression algorithm according to the classification scores of the respective regions, and the merged regions are classified using the trained classifier. Experiments demonstrate that, compared with the original SS, the improved SS algorithm increases the calculation speed by 13.8%, and its classification accuracy is 89.4%. Additionally, the classification algorithm for glass bottles has a certain resistance to noise.

Keywords: Classification of glass bottle, HBSN feature, improved selective search algorithm, LightGBM.

1 Introduction

Glass bottles have become a popular food packaging method for food manufacturers and consumers due to their portable, aesthetic, and non-toxic characteristics. However, the production of glass bottles is accompanied by a large amount of resource consumption. By classifying and recycling glass bottles, energy consumption, air pollution, mineral waste, and water consumption can be reduced by 32%, 20%, 50%, and 50%, respectively [Xia (1992)]. However, there are differences in the types, shapes, and sizes of glass

[1] School of Computer Science, Northeast Electric Power University, Jilin, 132000, China.

[2] Information and Communication Company, State Grid Tianjin Electric Power Company, Tianjin, 300000, China.

[3] Graduate School of Science and Engineering, Iwate University, Morioka, 020-8550, Japan.

[*] Corresponding Author: Shuqiang Guo. Email: guoshuqiang@gmail.com.

Received: 06 February 2020; Accepted: 08 March 2020.

bottles used in food packaging, which leads to difficulties in the classification and recycling of glass bottles, and ultimately a low reuse rate. Therefore, in the context of the current advocacy of garbage classification and sustainable development, it is extremely important to find an accurate and efficient glass bottle classification method.

Object classification technology is indispensable in the classification of different kinds of glass bottles, and is currently a popular research topic in the computer vision field. Some of the existing object classification algorithms exhibit good performance in the application of real scenes. Nassih et al. [Nassih, Amine and Hmina (2019)] combined DCT (discrete cosine transform) and HOG (histogram of oriented gradient), then used a back-propagation neural network (BPNN) [Zhong, Liu, Sun et al. (2018)] to classify faces; the accuracies on the BOSS and MIT databases were found to be 99.4% and 98.03%, respectively. Sevcan et al. [Sevcan and Hamidullah (2018)] proposed the use of LBPs (local binary patterns) to extract features from gastric cancer images, the use of the classical MDS (multidimensional scaling) algorithm to reduce the dimension of features, and finally the use of ANN to classify low-dimensional vectors to achieve early gastric cancer screening. Wei et al. [Wei, Tian, Guo et al. (2019)] used the Haar feature to extract the region of interest, then used the HOG algorithm to scan the region of interest to form feature vectors, and finally input the feature vectors into an SVM classifier for classification to achieve multi-vehicle target detection and tracking. Li et al. [Li, Niu, Fang et al. (2018)] proposed the use of the HOG algorithm to extract peanut features and the SVM algorithm to classify feature vectors to achieve the classification of a variety of peanuts; however, the problem of repeated calculation due to overlapping regions emerged in the HOG feature extraction window. To solve this problem, Huang et al. [Huang, Gu and Yang (2009)] introduced the idea of an integral graph to respectively improve the Haar and HOG features, and achieved good performance in practice.

The main concept of these classification algorithms is to use one or more feature descriptors to extract features from images, and then encode these features to form feature vectors. Finally, classifiers in machine learning are used to classify the feature vectors. However, when classifying objects, it is necessary to scan multiple images with different scales using sliding windows [Huang, Ren and Tan (2014)], and then judge the type of objects in each window. This exhaustive algorithm may exist in the target window, which results in a great time overhead. Therefore, Uijlings et al. [Uijlings, Sande, Gevers et al. (2013)] proposed a selective search (SS) algorithm to improve it. It uses an image segmentation algorithm to divide the image into independent color blocks, taking full account of the textures, colors, sizes, and overlap, and finally combines the segmented color blocks by combining multiple similarities to form a new image set. This algorithm has a high recall rate and can provide a reasonable target area for image classification. It solves the problem of the large time overhead caused by exhausting the target area in sliding windows, and is widely used in Faster R-CNN [Ren, He and Girshick (2015)]. Li et al. [Li, Zhou and Lu (2018)] applied the SS algorithm to vehicle face component detection. The detection algorithm first uses the HOG descriptor instead of the SIFT [Iyad and Sahar (2017)] descriptor to form texture features to improve the SS algorithm, and the improved algorithm is then used to segment different parts of the car face; it was found to achieve good segmentation results on the car face data set. Wu et al. [Wu and Zhan (2017)] proposed the use of the perceptual hashing algorithm and Hamming distance algorithm to

improve the similarity calculation in an SS algorithm. Experiments demonstrated that the improved algorithm exhibited obvious advantages over the sliding window on the LFW data set. These algorithms primarily aim at the improvement of similarity calculation in the SS algorithm, and use a new feature extraction method instead of the SIFT algorithm to achieve fast calculation. However, when calculating the texture similarity of different regions, the overlapping regions result in repeated computation.

In view of these problems, the improvement of the SS is first proposed in this paper. Moreover, different classifiers and the improved SS algorithm are selected to classify glass bottles. Finally, the superior classification algorithm of glass bottles is determined by comparing the performances of different classification algorithms on a data set of glass bottles. In the experimental stage, the original and improved SS algorithms are compared to verify the performance of the improved algorithm. Then, different glass bottle classification algorithms are tested on the data set to compare the effectiveness of the proposed algorithm. Finally, the stability of the glass bottle classification algorithm is tested by adding salt and pepper noise.

2 Improve selection search algorithm

The SS algorithm, Uijlings et al. [Uijlings, Sande, Gevers et al. (2013)] uses an image segmentation algorithm to divide an image into independent image regions, and uses a variety of similarity rules to merge the segmented image region to form new image regions. The specific process is as follows.

Step 1: Input a colorful image A.

Step 2: The algorithm proposed by Felzenszwalb et al. [Felzenszwalb and Huttenlocher (2004)] is used to segment the image A into different regions. The segmented image regions are recorded as set R, and $R = \{r_1, r_2, r_3, \ldots, r_i, \ldots, r_n\}$.

Step 3: Initialize the similarity set S to be empty, calculate the similarity $s(r_i, r_j)$ for any two elements r_i and r_j in R, and save the result $s(r_i, r_j)$ into the set S, that is, $S=S\cup s(r_i, r_j)$.

Step 4: If set S is not an empty set, iterate through set S to take out the maximum of $s(r_k, r_l)$ and merge the regions r_k and r_l to form a new image region r_t, that is, $r_t=r_k\cup r_l$. Remove the similarity $s(r_k, r_*)$ with r_k and the similarity $s(r_*,r_l)$ with r_l from set S. Compute the similarity set S_t corresponding to r_t. Merge S_t into S, that is, $S=S\cup S_t$. Merge image region r_t into set R, that is, $R=R\cup r_t$.

Step 5: If S is an empty set, output all image regions in R.

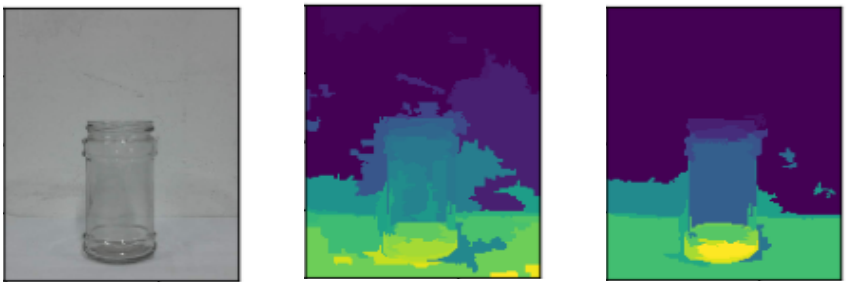
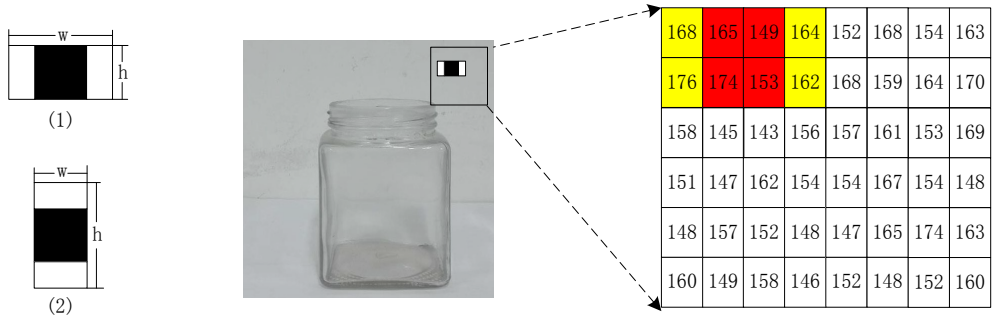


Figure 1: Effect of glass bottle segmentation

The segmented result after changing the segmentation threshold of the segmentation algorithm [Felzenszwalb and Huttenlocher (2004)] is presented in Fig. 1.

2.1 HBSN feature

The SS algorithm uses the SIFT feature [Ly, Teng and Lu (2016)] to extract texture features in the calculation of texture similarity. SIFT uses the Gauss differential to create a feature histogram on the R, G, and B channels corresponding to the image, which results in a large time consumption. To solve this problem, the HBSN feature (Haar-like based on SPP-Net) is proposed in this paper. First, the Haar-like descriptor is used to extract the features to generate the map of feature values from the input image. Many calculations of Haar-like operators are linear operations, which improves the operation speed. In addition, the map of feature values is mapped to a fixed-length feature histogram by using SPP-Net [He, Zhang and Ren (2014)]. Finally, the two processes are accelerated by using an integral graph to realize the fast calculation of texture similarity. As shown in Fig. 2(c), the horizontal direction is the *x*-axis, the vertical direction is the *y*-axis, and the unit of length is the pixel.



(a)Two descriptors (b) The input image (c) The descriptor scan the input images

Figure 2: The feature extraction of Harr-like

2.1.1 The map of feature values

The Haar-like feature descriptor is an algorithm for the extraction of facial features proposed by Viola et al. [Viola and Jones (2001)]. It exhibits a strong texture description ability by comparing the gray differences of different parts. The Haar-like feature is used as a texture feature to detect auto [Wei, Tian, Guo et al. (2019)], and has many kinds of descriptors. In this paper, two kinds of Haar-like descriptors are selected, as presented in Fig. 2(a). The corresponding sizes of the two descriptors are respectively 6×3 and 3×6, and Fig. 2(b) presents the input image. Fig. 2(c) presents the use of the two descriptors to extract features from the input picture, and the process of image feature extraction is as follows.

Step 1: As shown in Fig. 2(c), the Haar-like descriptor moves from the left-most end of the input image to the right-most end of the input graph with the stride of a pixel. In the process of Haar-like descriptor movement, Eq. (1) is used to calculate feature value V for the region covered by the Haar-like descriptor:

$$V = \sum_{p(x,y)\in Y} pixel(x,y) - \sum_{p(x,y)\in R} pixel(x,y) \tag{1}$$

where $p(x,y)$ represents the position of each pixel, *pixel* (x, y) represents the corresponding pixel value of each pixel, and Y and R respectively represent the yellow and red regions corresponding to Fig. 2(c).

Step 2: When the right end of the Haar-like descriptor reaches the end of the image, the Haar-like descriptor moves down one pixel.

Step 3: Repeat Steps 1 and 2 until the right end of the Haar-like descriptor reaches the lower right corner of the input image.

Step 4: The two kinds of Haar-like descriptors are used to output two maps of feature values corresponding to the input image. The size calculation of the map of feature values is given by Eq. (2):

$$W_1 = \frac{W-w}{s_x} + 1, \quad H_1 = \frac{H-h}{s_y} + 1 \tag{2}$$

where s_x represents the stride of the Haar-like descriptor moving along the x-axis, and s_y represents the stride of the Haar-like descriptor moving along the y-axis. Additionally, W, H, w, and h are shown in Fig. 2.

2.1.2 HBSN vector

The SPP-Net algorithm put forward by He et al. [He, Zhang and Ren (2014)] is used to solve the problem of the output of fixed-sized images from the input of images of different sizes. SPP-Net can map images of different sizes to vectors of fixed length, and has shown extremely good performance in neural networks.

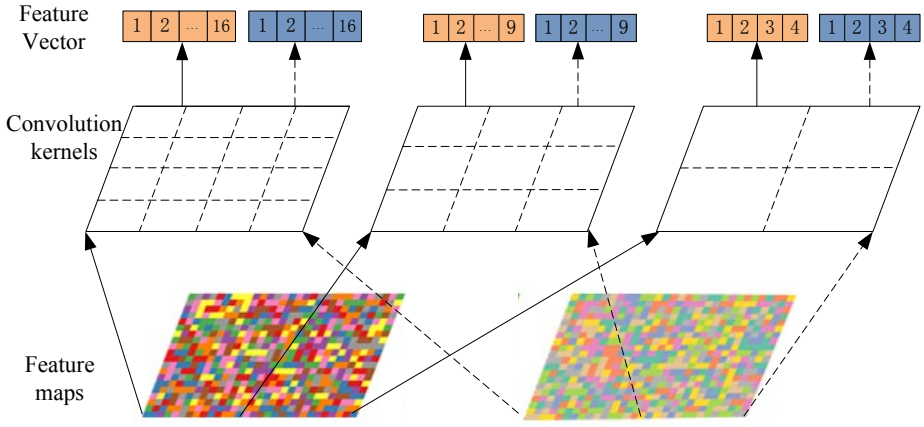


Figure 3: The formation of HBSN vectors

As presented in Fig. 3, SPP-Net is adopted, and the sizes of three convolution kernels are respectively 4×4, 3×3, and 2×2. The functions of these three convolution kernels are to divide the length and width of the map of feature values into four equal parts, three equal parts, and two equal parts, respectively. Each color block on the map of feature values in Fig. 3 represents one feature value.

Two maps formed by using two kinds of Haar-like descriptors respectively use the three convolution kernels. Thus, each map of feature values is divided into 16, 9, and 4 blocks. Eq. (3) is then used to pool the mean of each block in the corresponding region of the map of feature values and obtain the eigenvalue V_j corresponding to each block region. Thus, each map of feature values is mapped to a 29-dimensional eigenvector. According to the order in which different maps of feature values form eigenvectors under the same convolution kernel, two maps of feature values are mapped to a 58-dimensional vector, which acts as the eigenvector of every colorful channel of the input picture.

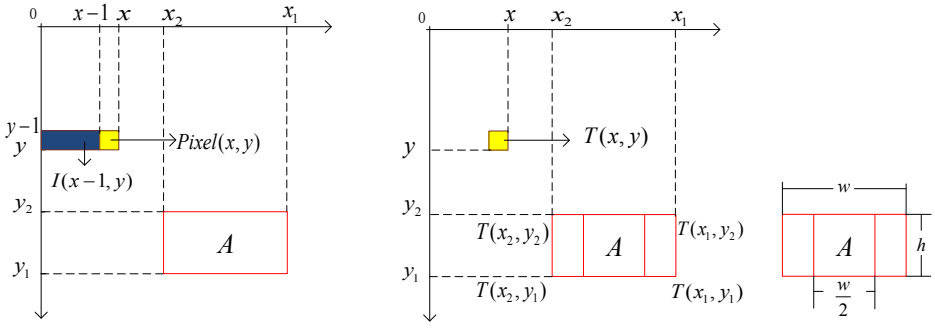
$$V_j = \frac{1}{N_j} \sum_{p(x,y) \in Block_j} pixel(x, y) \tag{3}$$

where $Block_j$ represents the j-th block region, $j \epsilon [1, 2, ..., 29]$, N_j represents the total number of pixels in the j-th block region, $p(x, y)$ represents the pixel in $Block_j$, and $pixel(x, y)$ represents the corresponding pixel value of the $p(x, y)$ pixel.

2.1.3 Integral graph acceleration

The integral graph was introduced into the image field by Viola et al. [Viola and Jones (2001)], and realizes a fast algorithm for the calculation of the sum of pixels in a rectangular region of arbitrary size. By indexing the four values corresponding to the four vertices of a rectangle, the integral graph performs fixed operations instead of computing between the pixels in the rectangular region. The formulas for calculating rectangular regions of different sizes are fixed linear combinations of four values corresponding to the four vertices. These linear operations greatly reduce the time consumption and accelerate the image.

Accelerating the process of forming map of feature values by integral graph algorithm. As shown in Fig. 4(a), the pixel values in the input image are traversed. The Formula 4 is used to calculate the value T (x, y) corresponding to any point (x, y) on the pixel value integral graph and the formed pixel value integral graph is shown in Fig. 4(b).



(a) The input image (b) The integral pixel value graph

Figure 4: Acceleration of HBSN vector

In Fig. 4(a), the sum of the pixel values in rectangular area A of any size is calculated, which is equivalent to using formula 5 to calculate T_R at the corresponding position of A

in Fig. 4(b).

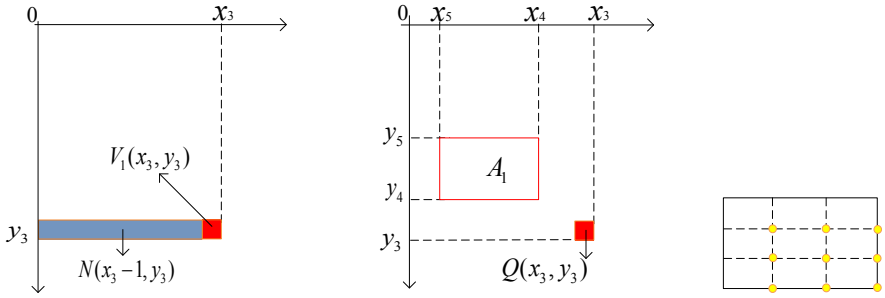
$$I(x,y)=I(x,y-1)+pixel(x,y)$$
$$T(x,y)=T(x-1,y)+I(x,y) \quad (4)$$

where $I(x, y)$ denotes the sum of pixel values from the point $(0, y)$ along the x-axis to the point (x, y), and $T(x, y)$ denotes the sum of pixels values in the rectangular region from the point (0, 0) to the point (x, y), where $I(-1, y) = 0$, $T(x, -1) = 0$.

$$T_R=T(x_1,y_1)-T(x_2,y_1)-T(x_1,y_2)+T(x_2,y_2) \quad (5)$$

where (x_1, y_1), (x_2, y_1), (x_1, y_2), and (x_2, y_2) respectively represent the four vertices of the rectangular region; $x_1 > x_2$ and $y_1 > y_2$.

When the rectangular box A composed of x_1, x_2, y_1, and y_2 is the Haar-like descriptor shown in Fig. 2(a), x_1, x_2, x_3, y_1, y_2, and y_3 satisfy Eq. (6) in the process of using the descriptor to form the feature value $V_1(x_3, y_3)$ in Fig. 5(a).



(a) The map of feature values (b) The integral map of feature values (c) The convolution of 3×3 process the region of A_1

Figure 5: Accelerate the texture vectors

When the Haar-like descriptor is the type (1) in Fig. 2(a), the rectangular region composed of x_1, x_2, y_1, and y_2 is divided into three rectangles. The coordinates of the three rectangles corresponding to eight points are (x_1, y_1), $(x_1\text{-}w/4, y_1)$, $(x_1\text{-}3\times w, y_1)$, (x_2, y_1), (x_1, y_2), $(x_1\text{-}w/4, y_2)$, and $(x_1\text{-}3\times w, y_2)$. Using Eqs. (5) and (1), the equation for calculating eigenvalue $V_1(x_3, y_3)$ is obtained as given by Eq. (7). Similarly, when the Haar-like descriptor is the type (1) in Fig. 2(a), the equation for calculating $V_2(x_3, y_3)$ is given by Eq. (8). Finally, the map of feature values corresponding to different Haar-like descriptors is obtained, as presented in Fig. 5(a).

$$x_1=s_x(x_3-1)+w, x_2=s_x(x_3-1)$$
$$y_1=s_y(y_3-1)+h, y_2=s_y(y_3-1) \quad (6)$$

where w and h respectively represent the width and height of the Haar-like descriptor, and s_x and s_y respectively represent the stride of the Haar-like descriptor on the x-axis and y-axis of the image.

$$V_1(x_3,y_3)=T(x_1,y_1)+2T(x_1-w/4,y_2)+2T(x_1-3w/4,y_1)+T(x_2,y_2)$$
$$-T(x_1,y_2)-2T(x_1-w/4,y_1)-2T(x_1-3w/4,y_2)-T(x_2,y_1) \quad (7)$$

$$V_2(x_3,y_3)=T(x_1,y_1)+2T(x_2,y_1-h/4)+2T(x_1,y_1-3h/4)+T(x_2,y_2)$$
$$-T(x_2,y_1)-2T(x_1,y_1-h/4)-2T(x_2,y_2-3h/4)-T(x_1,y_2) \quad (8)$$

where w and h respectively represent the width and height of the Haar-like descriptor.

Integral graphs are used to accelerate the process of mapping the feature values for texture vectors.

For the feature value $V(x_3, y_3)$ of any point (x_3, y_3) in Fig. 5(a), the value $Q(x_3, y_3)$ corresponding to (x_3, y_3) is calculated by Eq. (9) to form the integral map of feature values presented in Fig. 5(b). In this paper, s_x=1 and s_y=1 are used. When the input image is a rectangle A of any size in Fig. 4(a), Eq. (10) is used to calculate the coordinates of A in Fig. 5(b), and the area A_1 corresponding to area A is obtained. As shown in Fig. 5(c), area A_1 is divided into nine blocks by using the 3×3 convolution kernel.

$$N(x_3,y_3)=N(x_3,y_3-1)+V(x_3,y_3)$$
$$Q(x_3,y_3)=Q(x_3-1,y_3)+N(x_3,y_3) \quad (9)$$

where $N(x_3, y_3)$ represents the sum of the feature values from the point $(0, y)$ along the x-axis to the point (x_3, y_3), and $Q(x_3, y_3)$ represents the sum of the feature values in the rectangular region from the point (0, 0) to the point (x_3, y_3), where $N(-1, y_3)=0$, $T(x_3, -1)=0$.

$$x_4=x_1-w+1,\quad x_5=x_2+1$$
$$y_4=y_1-h+1,\quad y_5=y_5+1 \quad (10)$$

The process of calculating the mean of each block to form a dimension eigenvector value is equivalent to summing all feature values in each block and dividing them by the total number of feature points in the corresponding block. In the preceding calculation process, the sum of feature values T_R in any rectangle can be calculated by using the four values of the corresponding region on the integral map of feature values. k=3 is substituted into Eq. (11), and the mean of each block in Fig. 4(e) is obtained, forming a 9-dimensional vector. Similarly, when using convolution kernels of sizes 4×4 and 2×2, for A, k=4 and k=2 are respectively brought into Eq. (11) to form corresponding 16- and 4-dimensional vectors. According to the order of eigenvalues in Fig. 3, two maps of feature values are mapped to a 58-dimensional vector.

$$V(l,m)=\frac{1}{N}[Q(x_5+al,y_5+bm)-Q(x_5+a(l-1),y_5+bm)$$
$$-Q(x_5+al,y_5+b(m-1)+Q(x_5+a(l-1),y_5+b(m-1)] \quad (11)$$

where $a=(x_4-x_5)\div k$, $b=(x_4-x_5)/k$ and $N=a\times b$. k is constant and $k\in[2, 3, 4]$. $1\le l\le k$ and $1\le m\le k$.

2.2 The similarity $S(r_i, r_j)$

2.2.1 Computation of texture similarity

Two kinds of Haar-like descriptors presented in Fig. 3(a) are used to extract features from every color channel of image region r_i. Two kinds of Haar-like descriptors form a 58-dimensional feature vector on every color channel using Eq. (11). Every 58-dimensional vector obtained is then normalized by using L_1-norm. The Haar-like features formed by R, G, and B channels are arranged sequentially to form the 174-dimensional vector T_i corresponding to the image region r_i, $T_i=\{t_i^1, t_i^2,\ldots, t_i^k,\ldots, t_i^{174}\}$. Similarly, the texture feature vector of the image region is T_j, $T_j=\{t_j^1, t_j^2,\ldots, t_j^k,\ldots, t_j^{174}\}$. For image regions r_i and r_j, Eq. (12) is used to calculate the similarity $S_{text}(r_i, r_j)$:

$$S_{text}(r_i, r_j) = \sum_{k=1}^{174} \min(t_i^k, t_j^k) \tag{12}$$

2.2.2 Computation of color similarity

Every color channel in the image region r_i is divided into 25 groups, forming 25 *bins*. L_1-norm is used to regularize the values of the 25 *bins*. A total of 75 *bins* are formed in the R, G, and B color channels. The 75 *bins* are mapped to a 75-dimensional vector C_i, and $C_i=\{c_i^1, c_i^2,\ldots, c_i^k,\ldots, c_i^{75}\}$. Similarly, the color feature vector of the image region is C_j, $C_j=\{c_j^1, c_j^2,\ldots, c_j^k,\ldots, c_j^{75}\}$. The color similarity $S_{color}(r_i, r_j)$ of the images r_i and r_j is calculated by Eq. (13):

$$S_{color}(r_i, r_j) = \sum_{k=1}^{75} \min(c_i^k, c_j^k) \tag{13}$$

2.2.3 Computation of size similarity

To obtain as many candidate regions as possible, the smaller regions are merged first. The size similarity $S_{size}(r_i, r_j)$ between the image regions r_i and r_j is computed by Eq. (14):

$$S_{size}(r_i, r_j) = 1 - \frac{size(r_i) + size(r_j)}{size(im)} \tag{14}$$

where $size(im)$ denotes the size of the image where r_i and r_j are located, and the unit is pixels.

2.2.4 Shape compatibility

$S_{fill}(r_i, r_j)$ is defined to indicate the matching degree between the image regions r_i and r_j. If r_i is a part of r_j, r_i and r_j are merged; if there is no intersection between r_i and r_j, they are not merged. The formula of $S_{fill}(r_i, r_j)$ is as follows:

$$S_{fill}(r_i, r_j) = 1 - \frac{size(BB_{ij}) - size(r_i) - size(r_j)}{size(im)} \tag{15}$$

where $size(BB_{ij})$ denotes the size of the bounding box that is around r_i and r_j.

2.2.5 Computation of similarity S(ri, rj)

Using the linear combination of the four similarities mentioned previously, the calculation formula of $S(r_i, r_j)$ is as follows:

$$S(r_i, r_j)=a_1 S_{color}(r_i, r_j) + a_2 S_{texture}(r_i, r_j) + a_3 S_{size}(r_i, r_j) + a_4 S_{fill}(r_i, r_j) \tag{16}$$

where a_1, a_2, a_3, $a_4 \in [0, 1]$.

Finally, the image region r_t is produced by the combination of r_i and r_j. The $size(r_t)$, texture feature vector T_t, and color feature vector C_t corresponding to image region r_t are respectively calculated as follows:

$$size(r_t) = size(r_i) + size(r_j) \tag{17}$$

$$T_t = \frac{size(r_i) \times T_i + size(r_j) \times T_j}{size(r_i) + size(r_j)} \tag{18}$$

$$C_t = \frac{size(r_i) \times C_i + size(r_j) \times C_j}{size(r_i) + size(r_j)} \tag{19}$$

3.1 Data set

The size of positive and negative samples in the whole data set was 200×200 pixels. As presented in Fig. 5, samples were made by considering factors including different placement angles, illumination intensities, and glass bottle proportions. Positive samples were divided into five categories, there were 500 training samples and 300 test samples in each class, and there were 4000 pictures in the data set.

Figure 5: Glass bottle data set

3.2 Comparison between improved selective search and original SS algorithm

3.2.1 Speed comparison of candidate regions generation

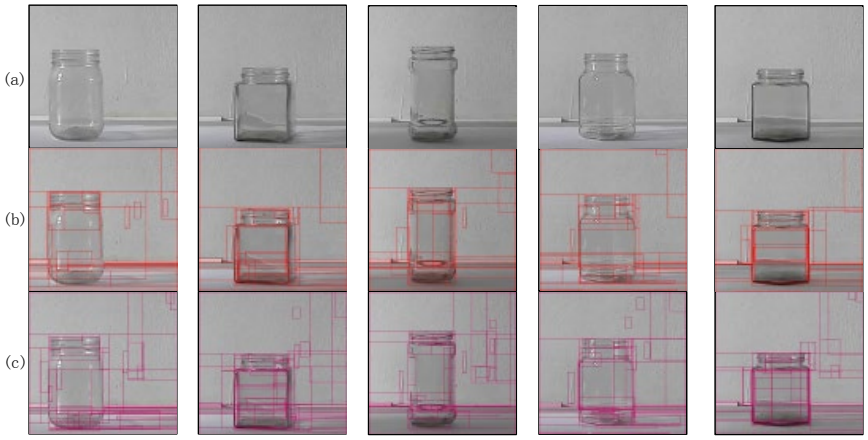


Figure 6: Comparison of candidate regions

The image in Fig. 6(a) presents the effect image after using the original SS algorithm, and the image in Fig. 6(b) presents the effect image after using the improved algorithm. The image segmentation algorithm [Felzenszwalb and Huttenlocher (2004)] used the values of sigma=0.8, scale=300, and min_size=800. The original and improved SS algorithms were respectively used for the image in the test set. For every class of glass bottle image, the average number of candidate regions generated per image and the corresponding average time consumption were tested. The data obtained are exhibited in Tab. 1.

Table 1: Comparison of time and regions

	Oriented Selective Search		Improved Selective Search	
class	Regions	Time(s)	Regions	Time(s)
1	132	0.828	147	0.714
2	127	0.824	141	0.709
3	119	0.703	136	0.617
4	134	0.797	143	0.677
5	124	0.704	139	0.609

As revealed by Tab. 1, the number of candidate regions generated by the improved SS algorithm on every class of glass bottle was significantly greater than that generated by the original algorithm, and the corresponding time consumptions were lower than those of the original algorithm. The experiments demonstrate that the speed of the generation of candidate regions by the improved algorithm was 13.8% faster than that by the original algorithm.

3.2.2 Comparison of overlap

In this study, ABO (average best overlap) [Uijlings, Sande and Gevers (2013)] was selected as the evaluation index of algorithm performance. For each fixed category c in the test set, each sample i in this type of glass bottle was calibrated manually, and the calibrated area is

represented by g_i^c. The improved SS algorithm and the original algorithm were used to process the sample i, and the corresponding candidate region sets L_1 and L_0 were obtained. The *ABO_ISS* of the improved SS algorithm is given by Eq. (20), and the corresponding *ABO_ISS* of the original algorithm is given by Eq. (21). The calculation of $Overlap(g_i^c, l_j)$ is given by Eq. (22). The corresponding data of the *ABO_ISS* and *ABO_OSS* values for different types of glass bottles are presented in Fig. 7.

$$ABO\_ISS = \frac{1}{|G^c|} \sum_{g_i^c \in G^c} \max_{l_j \in L_1} (Overlap(g_i^c, l_j)) \tag{20}$$

$$ABO\_OSS = \frac{1}{|G^c|} \sum_{g_i^c \in G^c} \max_{l_j \in L_0} (Overlap(g_i^c, l_j) \tag{21}$$

where G^c represents all samples of glass bottles in class c, and l_j represents the j-th candidate region in the set of candidate regions.

$$Overlap(g_i^c, l_j) = \frac{area(g_i^c) \cap area(l_j)}{area(g_i^c) \cup area(l_j)} \tag{22}$$

where $area(g_i^c)$ denotes the area corresponding to g_i^c, and $area(l_j)$ denotes the area corresponding to l_j in units of pixels.

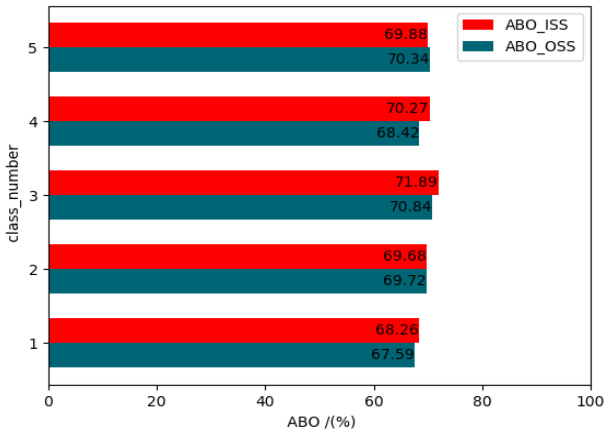


Figure 7: Comparison of *ABO*

In Fig. 7, the abscissa is the percentage of ABO, and the ordinate is the category of glass bottles. The length of the red bar corresponds to the ABO value of the improved SS algorithm, and the length of the blue bar corresponds to the ABO value of the original SS algorithm. As is evident, in terms of the ABO of Class 5, the performance of the original algorithm was better than that of the improved SS algorithm. In terms of the other four ABO values, however, the improved SS algorithm exhibited better performance. Overall, the improved SS algorithm was found to be superior to the original algorithm in ABO performance, and can provide a reasonable classification window for the subsequent classification of glass bottles.

3.3 Classification method and performance index of glass bottles

3.3.1 Classification algorithm for glass bottles

In this work, the improved SS algorithm and classifier were combined to classify glass bottles with various factors. The classification algorithm is presented in Fig. 8.

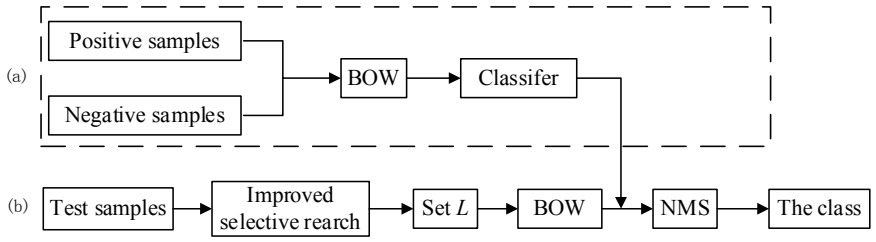


Figure 8: Classification algorithm for glass bottles

Training stage: As shown in Fig. 8, the target area of positive samples in the data set was calibrated, and the area with an overlap degree of ~25%-50% of the calibrated area of positive samples was regarded as negative samples. The BoW (Bag of Words) algorithm [Sreejini and Govindan (2019)] was used to extract features from positive and negative samples to form feature vectors. The feature vectors were then input into the classifier for training. In the process of iterative training, negative samples with high scores were added to train the classifier.

Testing stage: For the samples in the test data set, the improved SS algorithm was used to extract the candidate region set L. For the candidate region in L, the trained classifier was used to give the class score. The NMS (non-maximum suppression) algorithm [Vahid, Li, Jia et al. (2016)] was then used to merge the target region with a classification score greater than 0.7, and the merged region was input into the training. The trained classifier output the corresponding types of glass bottles and completed the classification of glass bottles.

The LightGBM [Ma, Sha, Wang et al. (2018); Zhao, Ye, Su et al. (2019], SVM [Sun, Lv, Mo et al. (2019); Czarnecki and Tabor (2015)], Bagging [Piero, Roozbeh and Enrico (2011)], Gaussian [Guo, Jia and Lyu (2019)], and Decision Tree [Ihssan, Amjed and Isam (2019)] algorithms have exhibited excellent performance in the fields of big data and data mining, and have also presented good classification effects on small data sets. Therefore, these five algorithms were chosen as the classifiers in glass bottle classification algorithms, and were then combined with the SS algorithm before and after improvement to construct different glass bottle classification algorithms. Finally, the performances of the different glass bottle classification algorithms were tested on data sets.

3.3.2 Performance indicators for classification of glass bottles

A (Accuracy), P (precision), R (recall), and F_1 [Uijlings, Sande and Gevers (2013)] were selected as the performance evaluation criteria of the glass bottle classification algorithms, and can respectively be calculated as follows:

$$A = \frac{TP + TN}{TP + TN + FP + FN} \tag{23}$$

$$P = \frac{TP}{TP + FP} \tag{24}$$

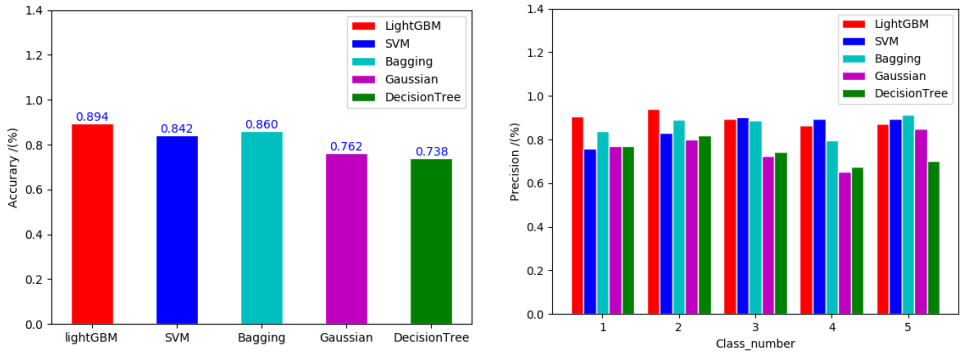
$$R = \frac{TP}{TP + FN} \tag{25}$$

$$F_1 = \frac{2TP}{2TP + FP + FN} \tag{26}$$

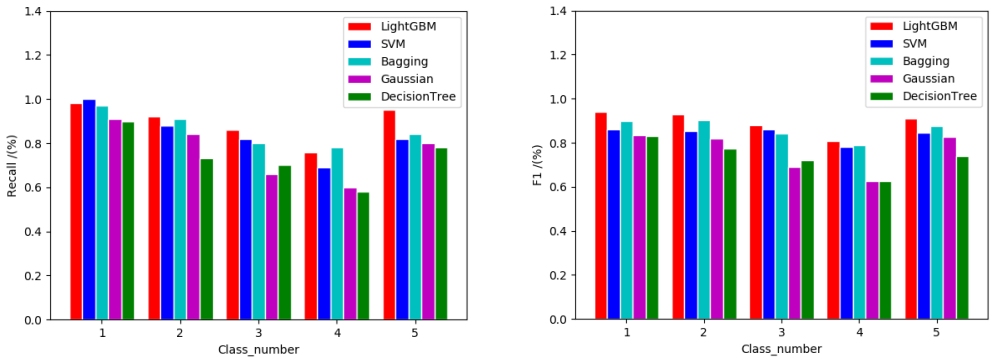
where TP represents the number of samples whose predicted class is positive and whose actual class is positive, TN represents the number of samples whose predicted class is negative and whose actual class is positive, FN represents the number of samples whose predicted class is negative and whose actual class is positive, and FP represents the number of samples whose predicted class is positive and whose actual class is negative. Among them, the positive class is the category of the specified class of glass bottles to be predicted, while the negative class is the type of glass bottles except the specific type of glass bottles.

Table 2: Performance comparison of classification algorithms

Algorithm	A	P	R	F_1
LightGBM+OSS	0.836	0.861	0.814	0.836
SVM+OSS	0.776	0.784	0.769	0.776
Bagging+OSS	0.825	0.832	0.814	0.823
Gaussian+OSS	0.724	0.732	0.716	0.724
DecisionTree+OSS	0.681	0.702	0.674	0.688
LightGBM+ISS	0.894	0.903	0.872	0.887
SVM+ISS	0.842	0.851	0.821	0.835
Bagging+ISS	0.860	0.865	0.842	0.853
Gaussian+ISS	0.762	0.760	0.754	0.757
DecisionTree+ISS	0.738	0.742	0.729	0.735



(a) Accurary (b) Precision



(c) Recall (d) F_1

Figure 9: Performance of different classification algorithms

In Tab. 2, OSS represents the original SS algorithm, and ISS represents the improved SS algorithm. For the same classifier, it can be seen that the improved SS algorithm was superior to the original algorithm in terms of the A, P, R, and F_1 values of glass bottle classification. Under the same combination of conditions, LightGBM was found to exhibit better classification performance than the other classifiers, and its corresponding classification accuracy was 89.4%.

Table 3: F_1 values of classification algorithms

Class	LightGBM	SVM	Bagging	Gaussian	DecisionTree
1	0.942	0.862	0.898	0.835	0.829
2	0.929	0.854	0.901	0.820	0.772
3	0.878	0.859	0.842	0.691,	0.722
4	0.809	0.780	0.788	0.625	0.624
5	0.909	0.845	0.875	0.825	0.739

Fig. 9 presents the performance comparison of glass bottles classified by the improved SS algorithm combined with the five classifiers. The A, P, R and F_1 values of each class of glass bottles are respectively presented in Figs. 9(a)-9(d). It is clear that the improved SS algorithm combined with LightGBM, SVM, and Bagging performed better in the classification of glass bottles. The F_1 values of different classification algorithms are presented in Tab. 3, which reveals that the F_1 value of LightGBM for each kind of glass bottle was higher than that of the other classifiers; the highest value in the first category was 94.2%. The experiments demonstrated that the improved SS algorithm combined with LightGBM exhibited good performance in the classification of glass bottles.

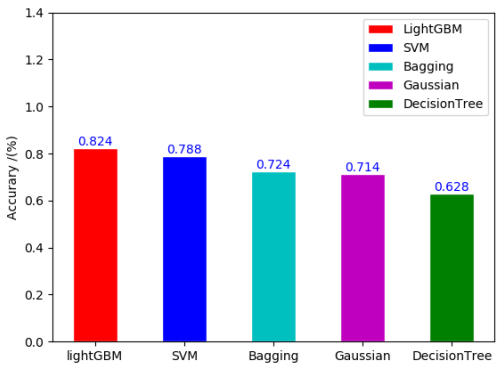
Figure 10: Samples of glass bottles images with noise

3.4 Noise testing of glass bottle classification algorithms

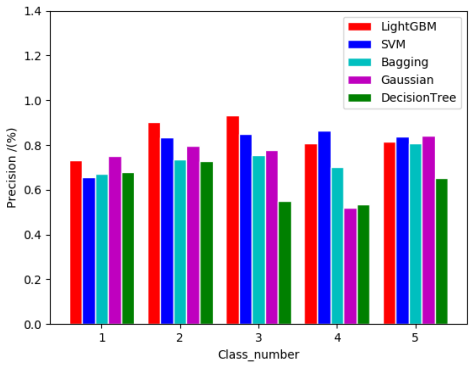
Noise was added to each picture in the test set. For each noise image, the amount of noise accounted for 1% of the total number of pixels in the image, and salt and pepper noise each accounted for 50% of the total amount of noise. Five kinds of glass bottle image samples with noise are presented in Fig. 10.

Five kinds of noise images were classified by using different glass bottle classification algorithms. As presented in Fig. 11(a), LightGBM exhibited the highest accuracy of 82.4% for glass bottle classification. In terms of precision performance, as presented in Fig. 11(b), LightGBM was found to have a good classification effect on the first three types of glass bottles.

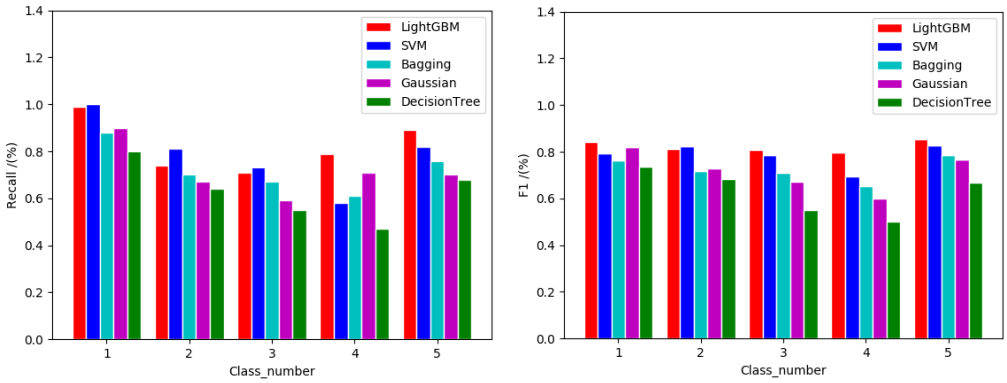
As exhibited in Tab. 4, the classification accuracy of the third kind of glass bottles was the highest at 93.4%. In the prediction of the latter two kinds of glass bottles, the precision of SVM was higher. In terms of recall, as presented in Fig. 11(c), SVM had a higher recall rate for the first three types of glass bottles, and the highest recall rate was 99.8% for class 1. LightGBM had higher recall rates for the latter two types of glass bottles. In terms of the F_1 value, LightGBM performed better than the other classification algorithms, as presented in Fig. 11(d). It is evident from the data in Tab. 4 that LightGBM had the highest F_1 values for the five samples as compared to the other classifiers, and the highest F_1 value for the fifth sample was 0.852. The experiments therefore demonstrate that LightGBM combined with the improved SS algorithm displayed the ability to resist noise in images.



(a) Accuracy



(b) Precision



(c) Recall (d) F_1

Figure 11: Performance of different classification algorithms under noise

Table 4: Performance of different classification algorithms under noise

Class	ISS+ Bagging+Noise			ISS+SVM+Noise			ISS+LightGBM+Noise		
	P	*R*	F_1	*P*	*R*	F_1	*P*	*R*	F_1
1	0.672	0.881	0.762	0.658	0.998	0.794	0.733	0.991	0.843
2	0.739	0.704	0.718	0.835	0.811	0.822	0.902	0.743	0.813
3	0.753	0.673	0.709	0.845	0.732	0.785	0.934	0.715	0.807
4	0.701	0.611	0.652	0.866	0.584	0.695	0.806	0.792	0.798
5	0.809	0.762	0.784	0.837	0.822	0.828	0.817	0.890	0.852

4 Conclusion

In this work, the selective search algorithm was first improved. The subsequent experiments demonstrated that the improved algorithm was superior to the original algorithm in terms of the number, speed, and overlap of candidate boxes. Moreover, the improved SS algorithm was applied to the classification of glass bottles via combination with different classifiers to obtain different glass bottle classification algorithms. The experiments revealed that the classifiers combined with the improved SS algorithm were superior to those combined with the original SS algorithm. Among them, the improved SS algorithm combined with the LightGBM classifier was found to be superior to the other glass bottle classification algorithms in terms of classification performance and noise resistance. In summary, the glass bottle classification algorithm constructed via a combination of the improved SS algorithm and LightGBM exhibits satisfactory classification performance for glass bottles with illumination changes and noise.

Funding Statement: This research is partially supported by:

(1) Research Foundation of Education Bureau of Jilin Province (JJKN20190710KJ).
(2) Science and Technology Innovation Development Plan Project of Jilin city (20190302202).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

Czarnecki, W.; Tabor, J. (2015): Multithreshold entropy linear classifier: theory and applications. *Expert Systems with Applications*, vol. 42, no. 13, pp. 5591-5606.

Felzenszwalb, P. F.; Huttenlocher, D. P. (2004): Efficient graph-based image segmentation. *International Journal of Computer Vision*, vol. 59, no. 2, pp. 167-181.

Guo, P.; Jia, Y.; Lyu, M. (2019): A study of regularized Gaussian classifier in high-dimension small sample set case based on MDL principle with application to spectrum recognition. *Pattern Recognition*, vol. 41, no. 9, pp. 2842-2854.

He, K.; Zhang, X.; Ren, S. (2014): Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 37, no. 9, pp. 1904-1916.

Huang, K.; Ren, W.; Tan, T. (2014): Overview of image object classification and detection algorithms. *Journal of Computer Science*, vol. 37, no. 06, pp. 1225-1240.

Huang, Q.; Gu, J.; Yang, W. (2009): Pedestrian detection based on gradient vector histogram. *Science and Technology and Engineering*, vol. 9, no. 13, pp. 3646-3651.

Ihssan, S.; Amjed, A.; Isam, A. (2019): Automated measurements of lumbar lordosis in T2-MR images using decision tree classifier and morphological image processing. *Engineering Science and Technology*, vol. 22, no. 4, pp. 1027-1034.

Iyad, A.; Sahar, A. (2017): Currency recognition using a smartphone: comparison between color SIFT and gray scale SIFT algorithms. *Journal of King Saud University—Computer and Information Sciences*, vol. 29, no. 4, pp. 484-492.

Li, X.; Zhou, Z.; Lu, S. (2018): Face component detection based on selective search algorithms. *Computer Engineering and Science*, vol. 40, no. 10, pp. 1829-1836.

Li, Z.; Niu, B.; Fang, P.; Li, G.; Yang, Z. et al. (2018): Classification of peanut images based on multi-features and SVM. *IFAC-PapersOnLine*, vol. 51, no. 17, pp. 726-731.

Lv, G.; Teng, S.; Lu, G. (2016): Enhancing SIFT-based image registration performance by building and selecting highly discriminating descriptors. *Pattern Recognition Letters*, vol. 84, pp. 156-162.

Ma, X.; Sha, J.; Wang, D.; Yu, Y.; Yang, Q. et al. (2018): Study on a prediction of P2P network loan default based on the machine learning LightGBM and XGboost algorithms according to different high dimensional data cleaning. *Electronic Commerce Research and Applications*, vol. 31, pp. 24-39.

Nassih, B.; Amine, A.; Ngadi, M.; Hmina, N. (2019): DCT and HOG feature sets combined with bpnn for efficient face classification. *Procedia Computer Science*, vol. 148, pp. 116-125.

Piero, B.; Roozbeh, R.; Enrico, Z. (2011): Bagged ensemble of fuzzy C-Means classifiers for nuclear transient identification. *Annals of Nuclear Energy*, vol. 38, no. 5, pp. 1161-1171.

Ren, S.; He, K.; Girshick, R. (2015): Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 39, no. 6, pp. 1137-1149.

Sevcan, A. K.; Hamidullah, B. (2018): Classification of molecular structure images by using ANN, RF, LBP, HOG, and size reduction methods for early stomach cancer detection. *Journal of Molecular Structure*, vol. 1156, pp. 255-263.

Sreejini, K. S.; Govindan, V. K. (2019): Retrieval of pathological retina images using Bag of Visual Words and pLSA model. *Engineering Science and Technology*, vol. 22, no. 3, pp. 777-785.

Sun, H.; Lv, G.; Mo, J.; Lv, X.; Du, G. et al. (2019): Application of KPCA combined with SVM in Raman spectral discrimination. *Optik*, vol. 184, pp. 214-219.

Uijlings, J.; Sande, K.; Gevers, T.; Smeulders, A. (2013): Selective search for object recognition. *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154-171.

Vahid, Z.; Li, S.; Jia, H.; Yong, X. (2016): Identifying the crack path for the phase field approach to fracture with non-maximum suppression. *Computer Methods in Applied Mechanics and Engineering*, vol. 312, pp. 304-321.

Viola, P. A.; Jones, M. J. (2001): Rapid object detection using a boosted cascade of simple features. *IEEE Computer Society Conference on Computer Vision & Pattern Recognition*.

Wei, Y.; Tian, Q.; Guo, J.; Huang, W.; Cao, J. (2019): Multi-vehicle detection algorithm through combining Harr and HOG features. *Mathematics and Computers in Simulation*, vol. 155, pp. 130-145.

Wu, S.; Zhan, Y. (2017): Face detection based on selective search and convolutional neural network. *Computer Applied Research*, vol. 34, no. 9, pp. 2854-2876.

Xia. T. (1992): Recycling and reuse of glass bottles and cans. *China Packaging*, no. 4, pp. 54-57.

Zhao, Q.; Ye, Z.; Su, Y.; Ouyang, D. (2019): Predicting complexation performance between cyclodextrins and guest molecules by integrated machine learning and molecular modeling techniques. *Acta Pharmaceutica Sinica B*, vol. 9, no. 6, pp. 24-39.

Zhong, J.; Liu, L.; Sun, Q.; Wang, X. (2018): Prediction of photovoltaic power generation based on general regression and back propagation neural network. *Energy Procedia*, vol. 152, pp. 1224-1229.