PANORAMA SCHOLARLY GROUP

Journal of Engineering Systems and Applications (JESA)



Automatic recognition of car models - based on Matlab

Zhihao Weng[a]*, Jian Yu[a], Yaqi Shen[b]

[a] *Zhejiang Jinhua Technology & Trade Polytechnic*

[b] *Zhejiang Gongshang University*

ARTICLE INFO

Keywords:
Vehicle type recognition; feature extraction; BP neural network

DOI: *https://doi.org/10.63802/jesa.v1.i1.78*

ABSTRACT

Based on vehicle classification standards, the construction of autonomous driving modes requires collecting relevant information, including parameters related to vehicle types. Subsequently, this mode can automatically classify vehicles (i.e., vehicle type identification), serving as the foundation for autonomous driving while providing data support for subsequent manual inspections and statistical analysis. The system achieves self-identification of vehicle categories through applications of image processing, image interpretation, and pattern recognition. The main research focuses are on three aspects: image preprocessing, feature extraction of vehicle characteristics, and vehicle type recognition based on BP neural networks. To enhance system accuracy and stability, we propose the following strategies: image enhancement, filtering, and edge detection. Features such as roof length ratio, roof height ratio, and front-rear ratio are extracted. A BP network model is constructed and BP algorithm is implemented. Through training and testing of eight samples from two categories, we have essentially completed preliminary classification of common vehicle types. Systematic testing confirms identification accuracy of 90% for both buses and sedans, demonstrating that our technical capabilities meet practical application requirements across all aspects.

∗ Corresponding author:Zhihao Weng.e-mail: gongyuxiru@foxmail.com

1. Introduction

1.1. Research Objectives and Significance

In the smart transportation system, vehicle monitoring plays a critical role. The intelligent transportation system (ITS), characterized by its core philosophy of maximizing the application of advanced technologies in current road environments to meet and optimize various transportation needs, employs diversified technological tools such as satellite positioning, autonomous driving guidance, traffic information dissemination, visual tracking, and computer management. These tools rapidly enhance road network efficiency, achieving safe and convenient transportation outcomes while forming an integrated road management strategy. Currently, intelligent transportation systems have become a global frontier in transportation studies, with numerous countries initiating research projects. These projects aim to address growing automotive traffic demands and ecological threats through information science, communication science, computer science, and intelligent management science, fundamentally transforming traditional transportation systems to improve resource utilization efficiency, system safety, while reducing resource consumption (particularly land use) and environmental pollution.

In intelligent transportation systems, highway automation serves as a cornerstone. The three key components of highway traffic automation management—site communication, remote vehicle monitoring, and toll collection at entry/exit points—require precise collection and processing of vehicle information and data, coupled with appropriate vehicle control, to ensure efficient toll operations and comprehensive highway automation. Vehicle detection technology plays an indispensable role in achieving automation for both highways and conventional roads.

Modern intelligent vehicles and outdated, complex road systems form the starting point of our research into intelligent transportation systems. The primary objective of this system is to expand and enhance current transportation network capabilities while achieving vehicle automation control. However, with the continuous emergence and application of new technologies, exploration of intelligent vehicles and their road systems has evolved into a comprehensive investigation of smart transportation systems. Today, intelligent transportation systems fully encompass the entire public transportation process, achieving the "intelligentization" inherent in their design.

1.2. Current Research Status Domestically and Internationally

To alleviate traffic congestion and enhance driving safety and ecological sustainability, countries have invested substantial resources and manpower into in-depth research to secure a dominant position in the future intelligent transportation system market. Since the 1970s, Europe has conducted extensive exploration of intelligent transportation systems. In 2000, the European Union initiated the e-Europe project to allocate more human, material, and financial resources to advance intelligent transportation systems in Europe. Japan pioneered research in road intelligence, successfully developing early road control systems in the 1970s. Today, Japan's intelligent transportation systems cover vehicle navigation, automatic tolling, and safe driving. Over decades, several countries including the United States and Japan have established leadership in intelligent transportation system exploration and innovation. Some U.S. local governments have implemented advanced autonomous driving technologies, reducing annual driving time and costs for Americans.

Our ancestors began integrating computer science and digital science into transportation in the 1970s, yet wisdom transportation theory was only proposed in the 1990s. In late 1970s and early 1980s, China first attempted to incorporate electronic information technology into traffic operations and management, conducting in-depth research on traffic signal control at intersections. During early 1990s, rapid economic growth in China increased transportation demands, while public infrastructure development lagged behind the rapid expansion of vehicle numbers. Data shows China currently has over 50 million vehicles, with annual urban vehicle growth rates of approximately 15%, while urban road construction grows by only 3%-5% annually. To address road shortages, rising vehicle numbers, and challenges in traffic safety and environmental protection, China is actively advancing infrastructure development and has initiated intelligent transportation system R&D projects. Following in-depth exchanges and collaborations with advanced nations, China has achieved significant progress in theoretical exploration, product development, and engineering applications. In recent years, numerous domestic universities and research institutions have begun exploring this field. According to published papers, institutions like Shanghai Jiao Tong University, Sichuan University, and Tongji University's Artificial Intelligence Institute are conducting relevant research. While advancing image processing and pattern recognition technologies, we continuously adopt new methodologies such as wavelet analysis, mathematical morphology methods, and various neural network techniques. Due to the complexity of this issue, precise accuracy, response time, and comprehensive resilience to external influences remain unachieved, along with universally accepted optimization strategies and holistic computational frameworks for such challenges.

1.3. Introduction to the Practical Application Software MATLAB

This paper utilizes MATLAB 23.2 as the software simulation environment, comprehensively simulating the image processing workflow and recognition methods for vehicle shapes. Through in-depth research on MATLAB's internal data structures and image types, we have established a solid foundation for implementing vehicle type recognition algorithms. By analyzing and comparing simulation results, we optimized the image processing workflow and recognition algorithms, ultimately achieving a satisfactory outcome. The simulation outcomes confirmed the practicality of image processing and recognition technologies.

MATLAB's name combines the first three characters of "Matrix" and "Laboratory." Dr. Cleve MOLER, a leader in the computer science department at the University of New Mexico, founded this project. While teaching linear algebra, he realized the difficulty of programming with other high-level languages and designed MATLAB. This software incorporated subroutines from the widely popular EISPACK (software package for eigenvalue computation) and LINPACK (linear algebra software package), built using the FORTRAN language, to create an interactive system integrating command translation and scientific computation. The second MATLAB version emerged in the early 1980s, entirely programmed in C, enabling MATLAB to perform not only numerical calculations but also data visualization through graphics.

As MATLAB's usage expanded, in 1984, Professor MOLER and a team of mathematicians and software professionals established MathWorks, a software development enterprise dedicated to MATLAB research and development. The MATLAB 4.0 version, released by MathWorks in 1992, marked a significant improvement. In 1993, a PC version of MATLAB was launched, enabling integration with Windows and broadening the system's application scope. MATLAB 5.0 was released in 1997, followed by upgrades to 5.2 and 5.3. The 6.x version debuted in 2003 with enhanced computational performance. Using relevant programming software, we can process images to achieve instant visual recognition, improving user experience and aligning better with user needs. Java interfaces have been added. Most toolboxes have been upgraded, benefiting both new and existing users. The MATLAB 2025a version has now been released.

We decided to use Matlab for simulation, and the reasons are as follows:

(1) Matlab is highly suitable for matrix operations, featuring extensive matrix functions, mathematical problem-solving capabilities, and related computational rules. Since images collected by data acquisition cards (A/D conversion) are typically stored in computers as matrices or arrays, using Matlab for image processing is highly convenient.

Matlab includes numerous neural network functions, whose stability and ease of use significantly reduce the time required for algorithm research. The BP neural network was applied to vehicle type identification in this paper due to its maturity in current technological development. This approach not only minimizes time consumption but also allows focus on algorithm optimization.

Matlab also provides basic image processing operations, such as data sorting, median filtering, grayscale transformation, and fundamental mathematical morphology operations, which can be rapidly invoked. Additionally, Matlab offers robust image visualization capabilities, creating an excellent debugging environment for this project: program optimization can be guided by visualized image content.

Despite Matlab's advantages in conciseness, convenience, and efficiency, its notable limitations in image processing include: - 'Low execution efficiency of programs generated from M-files' (many practical systems now use faster languages like C++ or VC).

- 'Generated code cannot operate independently in the Matlab environment'. These limitations can be mitigated using the Matlab Compiler, which translates programs (converted to function files) into executable code. The resulting outputs include: (1)'Generate C code and create a MEX file'. This approach excels when programs exhibit high vectorization levels or contain loop structures, accelerating processing efficiency. MEX files exist as ASCII code, offering strong privacy protection and resisting unauthorized modifications.

(2) 'Develop C or C++ code' that integrates with other C++ components to build standalone peripheral applications. These programs do not require the Matlab environment (but must rely on Matlab's C/C++ math libraries).

(3) 'Generate dynamic link libraries (DLLs) or C++ static libraries' without needing the Matlab environment, though Matlab's math libraries must still be supported.

Matlab contains a vast array of matrix computation methods. Additionally, the core of image processing lies in performing mathematical operations on images. Matlab demonstrates significant advantages in this field, as it can fully leverage its built-in function library during programming operations, thereby not only streamlining computational processes but also making programs more integrated. Before commencing coding, we need to develop a comprehensive plan and execute it through distinct modules and steps. The preparation steps prior to programming are outlined below:

(1) Determine the operational approach and format of image files, as well as the required dimensions of images to be displayed. Typically, we use JPEG format, though we opt for BMP images for analysis in certain operations. This is because BMP images retain richer color information compared to JPEG, and converting BMP to JPEG results in loss of color information. Regarding grayscale transformation, since the fundamental principles are well-established, no special functions need to be developed—direct utilization of Matlab's built-in functions for grayscale conversion suffices. Matlab's graphical capabilities are exceptional, making this process design remarkably efficient.

(2) Define the types and forms of required data, such as whether to employ matrix methods or data-point approaches. When processing automotive images, we must also consider data categorization. Generally, when conducting deeper calculations on images, the preferred method involves utilizing pixel data. For fundamental image computations, a matrix-based approach is adopted. For instance, in calculating image projections, we employ both matrix algorithms and data-point operations. Data-point operations in Matlab are exceptionally convenient, akin to array operations in C language. Furthermore, Matlab's programming language exhibits strong logical structure, with syntax very similar to conventional high-level programming languages, enabling execution of complex calculations.

(3) During programming, we must balance the use of MATLAB's built-in functions versus self-developed functions. This step is crucial; regardless of the state, functions must be created. However, MATLAB's built-in functions cannot adhere to any laws. Our perspective is that if we have deeply understood and mastered established, mature algorithms (e.g., median filtering algorithms), we should not waste time reimplementing them but directly utilize MATLAB's built-in functions. This approach both reduces research time consumption and saves significant effort. Conversely, when encountering algorithms requiring multiple programming tasks that MATLAB lacks pre-existing implementations, we must consider their special nature. In such scenarios, we must develop predefined functions. While MATLAB provides certain functions that must be utilized during program design, its internal functions fail to meet the required standards, necessitating manual creation of new functions.

(4) Adopting a modular programming design, all recognition steps for vehicle type identification are modularized. This makes subsequent program interpretation and modification extremely convenient.

1.4. Main Research Content of This Thesis

This paper starts from the process of vehicle type recognition and describes each step in detail in order.

The main chapters are arranged as follows:

Chapter 1: Introduction. This chapter elaborates the objectives and significance of the research topic, reviews domestic and international research status, and provides an overview of the practical application software MATLAB.

Chapter 2: Preprocessing of Vehicle Type Images. It first performs binary processing on images, then enhances and smoothes vehicle type images, and finally conducts edge detection.

Chapter 3: Feature Extraction. This chapter explains the methods for feature refinement and the computational approaches involved.

Chapter 4: Design of BP Neural Network Classifier. The characteristics of BP networks and their structural design are discussed, along with the implementation of BP network training algorithms.

Chapter 5: Analysis and Discussion of Experimental Results. The focus lies in evaluating experimental effectiveness, investigating potential flaws in the experimental process, and proposing optimization strategies.

Chapter 6: Conclusion. This chapter comprehensively summarizes the research findings and offers suggestions for future in-depth exploration.

The paper concludes with a section of References.

2. Preprocessing of Vehicle Type Images

2.1. Image Binarization

A binary image refers to an image where each pixel's gray level has only two possible values: either 0 or 255. This creates a distinct black-and-white contrast across the entire image. Through binarization, the image of the car's surroundings can be transformed into a two-dimensional graph with 0-1 parameters. To obtain an ideal binary image, threshold segmentation is commonly used. This method excels in processing images with high contrast. Its advantages include a simple computational process and the ability to identify non-overlapping regions using closed and continuous edges. Any image with a gray level exceeding or falling below the preset threshold is considered an object (marked as "255" representing 1), while those falling outside this range are excluded (marked as "0" representing background, also 0). Consequently, the object's edge becomes a collection of points where each point has at least one neighboring pixel that does not belong to the object. When the target object has a uniform gray level and resides in an environment with varying gray levels, thresholding yields optimal results. If the distinction between object and background lies in texture variations rather than gray-level differences, we can convert this attribute into a gray-level distinction and apply thresholding. To enhance segmentation stability and adaptability, the system should automatically set thresholds. Image segmentation algorithms based on object, environment, and application domain knowledge demonstrate superior generality and adaptability compared to fixed-threshold algorithms. This information encompasses the object's gray-level properties, size, proportion in the image, and the quantity of various objects. Here, the gray histogram represents a gray-level attribute typically used in image segmentation.

Global thresholding and local thresholding are the two types of threshold segmentation. Local thresholding involves dividing the original image into smaller segments and assigning appropriate thresholds to each. Post-thresholding, adjacent sub-images may exhibit gray-level discontinuities at their edges, requiring smoothing techniques to eliminate these. Common local thresholding methods include the gray difference histogram method and the differential histogram method. Although local thresholding improves segmentation efficiency, it has certain limitations:

(1) All sub-images must be of moderate size; otherwise, the resulting data will lack value.

(2) Sub-image partitioning is random. If a sub-image coincides with the target region or background region, statistically-based segmentation may lead to worse outcomes.

.(3) For each sub-image, the local thresholding method requires statistical processing, which is relatively slow and unsuitable for real-time requirements.

In the field of image processing, global thresholding segmentation technology is widely used. Its basic principle involves using a specific threshold to partition the entire image. The grayscale histogram is employed as the standard for threshold selection. Depending on the threshold selection method, it can be categorized into modal methods, iterative methods, and others. All these methods are based on the image histogram to determine the segmentation threshold. Additionally, there are other techniques such as between-class variance thresholding segmentation, two-dimensional maximum entropy segmentation, fuzzy thresholding segmentation, co-occurrence matrix segmentation, and region-growing methods.

In this paper, we employed binary thresholding as follows:

```
I = imread('1.jpg');      % Read the original image
figure, imshow(I);        % Display the original image
I2 = rgb2gray(I);         % Perform grayscale conversion
figure, imshow(I2);       % Display the grayscale image
```

2.2. Image Enhancement

In the fields of image recognition and image cognition, image enhancement is considered a primary processing method. Throughout the process of acquiring and transmitting image data, image quality may inevitably degrade. For instance, image edges may become

overly blurred to meet detection targets' size and shape requirements. In a highly satisfactory image, we can observe more unknown black and white dots distributed within it. Imaging distortions such as distortion, warping, etc., also exist. In short, due to factors such as visual experience and the convenience of recognition, output images face numerous potential challenges, which are categorized as "quality defects."

The significance of image enhancement lies in optimizing image quality through various methods or transforming them into formats more suitable for human and computer analysis. This primarily involves making appropriate adjustments to images based on requirements to highlight specific features, remove or reduce unnecessary information, such as adjusting brightness, eliminating noise, or enhancing edge processing. This section focuses on core technologies of image enhancement, including image transformation, histogram transformation, and image smoothing.

2.2.1. Image Transformation Enhancement

In the process of computer image processing, image transformation refers to a mathematical technique aimed at achieving specific objectives (typically extracting key information from images). The transformed image becomes more convenient, easier to process, and more manageable.

In practical applications, image transformation essentially constitutes a mathematical challenge of selecting an appropriate transformation kernel for the original image function. Fundamentally, image transformation possesses profound physical implications. Among all image transformation methods, the Fourier transform is the most widely applied. It can be regarded as an alternative representation of a function. Mastering the Fourier transform enables individuals to contemplate problem-solving strategies simultaneously in the spatial or frequency domain while handling complex issues in a concise manner. A single Fourier transform reveals the frequency distribution of a function within the system spectrum. Suppose we intend to perform specific operations within the frequency domain to adjust certain properties of the function (e.g., image enhancement). In such cases, a double Fourier transform would be employed. The Fourier transform encompasses two types of transformations:

(1)Continuous Fourier Transform

Assume that the function f(x) is a continuous function of the variable x and is absolutely integrable on (-, +), then the Fourier transform of $ f(x) $ is defined as follows:

$$F(u) = \int_{-\infty}^{+\infty} f(x) e^{-j2\pi ux} dx \qquad (2\text{-}1)$$

Assuming F(u) is integrable, the Fourier transform of f(x) is defined as follows:

$$F(x) = \int_{-\infty}^{+\infty} F(u) e^{j2\pi ux} du \qquad (2\text{-}2)$$

The Fourier transform pair is composed of equations (2-1) and (2-2). Before the Fourier transform, the variable domain x is referred to as the spatial domain, while the transformed variable domain u is called the frequency domain. In practical applications, we typically make relevant settings based on these two formulas.

The Fourier transform of a real function is usually a complex function. That is, F(u) can be expressed as:

$$F(u) = R(u) + jI(u)$$

In the formula

$$|F(u)| = \sqrt{R^2(u) + I^2(u)}$$

$$\theta(u) = \arctan[\frac{I(u)}{R(u)}]$$

The magnitude function is referred to as the Fourier spectrum $|F(u)|$, $\theta(u)$ *which represents the phase angle of the Fourier transform.* $F(x)$ *The square of the amplitude spectrum generally represents the energy spectrum, i.e.:*

$$|F(u)|^2 = R^2(u) + I^2(u)$$

The variable u *appearing in the Fourier transform is called the frequency variable. The origin of this name is as follows: using Euler's formula, the exponential* $e[-j2\pi ux]$ *can be expressed in the following form:*

$$e[-j2\pi ux] = e^{-j2\pi ux} = \cos(2\pi ux) - j\sin(2\pi ux)$$

If the integral in Equation (2-1) is expressed as the limit of a discrete sum, it is evident that F(u) is an infinite sum containing both sine and cosine terms. Moreover, each value of u determines the frequency of the corresponding sine-cosine pair.

The Fourier transform can be easily extended to two-dimensional cases, assuming the function $f(x, y)$ *is continuous and integrable, and* $F(u, v)$ *is integrable, then there exists the following Fourier transform pair:*

$$F(u,v) = \int_{-\infty}^{+\infty} f(x,y) e^{-j2\pi(ux+uy)} dxdy$$

$$f(x,y) = \int_{-\infty}^{+\infty} F(u,v) e^{j2\pi(ux+uy)} dudv$$

The two-dimensional Fourier transform of a function can also be written as the following form:

$$F(u,v) = R(u,v) + jI(u,v)$$

Amplitude spectrum is:

$$|F(u,v)| = \sqrt{R^2(u,v) + I^2(u,v)}$$

Phase angle:

$$\theta(u,v) = \arctan[\frac{I(u,v)}{R(u,v)}]$$

Spectral energy is:

$$|F(u,v)|^2 = R^2(u,v) + I^2(u,v)$$

2. Discrete Fourier Transform

Generally, the Fourier transforms we use on computers are presented in a discrete form, known as the Discrete Fourier Transform (DFT). Two key factors drive the DFT: first, both its input and output are discrete, making it simple for computers to process; second, the DFT has a fast algorithm—Fast Fourier Transform (FFT)—thereby enabling relatively easy computation.

Assume that the function $f(x)$ *is sampled at $ N $ evenly spaced intervals, yielding a discretized function* $f(m)$ $(1,2,\mathrm{L}\ ,N-1)$. *The one-dimensional discrete Fourier transform pair is defined as follows:*

$$F(p) = \frac{1}{n}\sum_{m=0}^{N-1} f(m)e^{-j2\pi pm/N} \qquad p = 0,1,2,\mathrm{L}\ ,N-1$$

$$f(m) = \sum_{m=0}^{N-1} F(p)e^{j2\pi pm/N} \qquad m = 0,1,2,\mathrm{L}\ ,N-1$$

Digital video is a two-dimensional discrete function. The following defines the relationship between the two-dimensional discrete Fourier transform (DFT) and the inverse DFT:

$$F(p,q) = \frac{1}{MN}\sum_{n=0}^{N-1}\sum_{m=0}^{M-1} f(m,n)e^{-j2\pi(pm/M+qn/N)} \qquad \begin{matrix} p = 0,1,2,\mathrm{L}\ ,M-1 \\ q = 0,1,2,\mathrm{L}\ ,N-1 \end{matrix}$$

$$f(m,n) = \sum_{p=0}^{M-1}\sum_{q=0}^{N-1} F(p,q)e^{j2\pi(pm/M+qn/N)}$$

$$p = 0,1,2,\mathrm{L}\ ,M-1$$

$$q = 0,1,2,\mathrm{L}\ ,N-1$$

2.2.2. Histogram Transformation Enhancement

1.grayscale histogram

A continuous image $f(x,y)$ *with area A becomes a digital image* $f(m,n)$ *with M rows and N columns after digitization. Generally, the number of pixels with different gray levels varies in the digital image* $f(m,n)$. *A histogram is a statistical chart used to represent the gray level distribution of an image. For the digital image* $f(m,n)$, *the horizontal axis represents the gray level r, and the vertical axis represents the probability value* $p(r_i)$ *of occurrence of this gray level.*

Let the gray level of a digital image pixel be $r_0, r_1, \mathrm{L}\ , r_{L-1},$, *then* $p(r_i)$ *is:*

$$p(r_i) = \frac{n(r_i)}{N} \qquad i = 0,1,\mathrm{L}\ ,L-1$$

Wherein, $n(r_i)$ *represents the total number of pixel points with gray level* r_i, *and N denotes the total number of pixel points in an image, with*

$$\sum_{i=0}^{L-1} p(r_i) = 1$$

Although the gray-level histogram cannot clearly display the positions of pixels at specific gray-levels nor intuitively present image information, statistical property-based histograms can reveal gray-level distribution characteristics of images, enabling us to obtain gray-level distribution concepts related to image quality, such as overall brightness, contrast, and object separability, which serve as primary references for processing approaches. In MATLAB, the imhist function performs histogram computation and description with the following syntax:

Imhist(I,N) - Displays the histogram of gray-level image I on a gray-level bar of length N. The default value of N is 256 for gray-level images and 2 for binary images.

mhist(X,MAP) - Displays the histogram of indexed image X on the color bar defined by colormap MAP, where the colormap must have at least the same length as the indices in X.

[counts,X]=imhist(...) - Returns the number of pixels at each gray-level for gray-level images, and the count of pixels corresponding to colormap entries for indexed images.

2. Histogram equalization

Another strategy for enhancing gray levels is to perform histogram equalization. Images with low contrast will have their histogram distribution concentrated within a smaller range. After homogenization, the histogram will exhibit uniform distribution across gray levels, where each gray level has an equal probability of occurrence. At this point, the image's entropy reaches its maximum value, indicating that the image contains the maximum amount of information. The following describes the operational process for histogram equalization:

(1)List the gray levels $r_i, i = 0,1\mathrm{L}\ , L-1$ *of the original image, where L is the number of gray levels.*

(2)Count the number of pixels ($n(r_i), i = 0,1\mathrm{L}\ , L-1$*) for each gray level.*

(3) Calculate the frequency $p(r_i) = n(r_i)/N, i = 0,1\mathrm{L}\ , L-1$ *of each gray level in the histogram of the original image, where N is the total number of pixels in the original image.*

(4) Calculate the cumulative distribution function $T(r_i) = \sum_{j=0}^{i} p(r_j), i = 0,1\mathrm{L}\ , L-1$

(5) Apply the following formula to calculate the grayscale level $g_i, i = 0,1\mathrm{L}\ , p-1$ *of the mapped output image,* p *which is the number of output grayscale levels:*

$$g_i = INT[(g_{\max} - g_{\min})T(r_i) + g_{\min} + 0.5]$$

Among them, INT is a rounding symbol.

(6) Calculate the number of pixels in the grayscale after mapping, denoted as $n(g_i), i = 0,1,\mathrm{L}\ , P-1$.

(7) Calculate the histogram $p(g_i) = n(g_i)/N, i = 0,1,\mathrm{L}\ , P-1$ *of the output image.*

(8) Adjust the gray level of the original image using the mapping relationship between r_i *and* g_i *to obtain an output image with a uniformly distributed histogram.*

In MATLAB, histogram homogenization can be achieved using the $histeq$ *-number. The calling method of the function is:*

[J,T]=$histeq$ *(I,N)*

2.2.3. Image smoothing

The main goal of image smoothing is to reduce the selection of objects (extracting a larger portion and removing a smaller portion) and reduce noise in the image. There are several ways to do it.

1.Domain averaging method

Domain averaging is a method of handling local spatial domains. Suppose a digital image $f(x,y)$ *is an array of* $M \times N$ *, and the smoothed image is* $g(x,y)$*. The gray level of each pixel is determined by the average gray level of several pixels in a predetermined field containing* (x,y)*, and the smoothed image is obtained using the following equation.*

$$g(x,y) = \frac{1}{K} \sum_{(i,j) \in s} f(i,j)$$

In the formula $x = 1,2,\mathrm{L}\ , M$ *,* $y = 1,2,\mathrm{L}\ , N$ *,* S *are the predetermined domain of* (x,y) *pixels (excluding* (x,y) *pixels), and K is the total number of coordinate points within S.*

Let noise $e(x,y)$ *be additive white noise with a mean of 0 and a variance of* σ^2*, and noise* $f(x,y)$ *is uncorrelated with each other. The image with noise interference is:*

$$g(x,y) = f(x,y) + e(x,y)$$

The image $g(x,y)$ *processed by the domain averaging method is:*

$$g(x,y) = \frac{1}{K} \sum_{(i,j)} \sum_{\in S} f(i,j) + \frac{1}{K} \sum_{(i,j)} \sum_{\in S} e(i,j)$$

The average value of residual noise after processing is:

$$E[\frac{1}{K} \sum_{(i,j)} \sum_{\in S} e(i,j)] = 0$$

The variance of residual noise is:

$$D[\frac{1}{K} \sum_{(i,j)} \sum_{\in S} e(i,j)] = \frac{1}{K} \sigma^2 = 0$$

After domain averaging, the variance of residual noise has been reduced to 1/K. However, after image conversion, the contour of the target object may become blurry or the detailed features may disappear.

2. Choose the average method

The basis for choosing the averaging method is the domain averaging method. This method only performs equal quantization on image elements with consistent or close grayscale values, or weights them based on specific grayscale levels of the image, avoiding unclear image boundaries. The typical value for mold size is. If the model specification is high, its denoising effect will be more prominent. However, due to the tedious calculation process and the large number of images that need to be processed, this may lead to subtle differences in the images being ignored, resulting in image blurring. When the denoising function of the template is not significant enough, we can perform multiple repeated operations while observing the final result. Threshold method and semi domain method are commonly used selection averaging methods.

1) Threshold method

The mathematical expression of the threshold method is as follows:

$$g(x,y)=\begin{cases}\dfrac{1}{M}\sum_{(i,j)\in S} f(x,y)\\ f(x,y)\end{cases}$$

$$\left|f(x,y)-\frac{1}{M}\sum_{(i,j)\in S} f(x,y)\right|>T$$

其他

Here, S represents the area of a point, M represents the total number of pixels in that area, and T represents a pre-set non negative threshold. Once the deviation between the values of certain image elements and their corresponding regions exceeds the preset threshold T, the grayscale value of that image element will be maintained. Compared to the domain average method, the image that has been smoothed has a reduced degree of blur. If there is a significant difference between the grayscale value of a pixel and the grayscale value of the pixel in its region, then it is definitely noise. Therefore, we can choose to use the average value of the region instead of the grayscale value of this pixel, which can still maintain good smoothness.

2) Semi domain law

The specific algorithm of the image semi domain method is set within the local area of window 3×3

Sort by A_i*, with the top 5 points with higher grayscale values forming Group B, and the bottom 3 points with lower grayscale values forming Group A;*

Set threshold value T;

Find the average values of $\overline{A}$ *and* $\overline{B}$ *for groups A and B.*

If $\left|\overline{A}-\overline{B}\le T\right|$ *considers no edge passing, use the average of 8 domains; On the contrary, it is considered that there are edges passing through, and the 5 points in P and B are averaged by 6 points.*

3. Median filtering

Median filtering is a non-linear filtering method that does not rely on the statistical properties of the image, making it relatively simple in practical applications. This is an image-based characteristic where noise is typically presented in the form of independent points, which correspond to a small number of pixels, while images are composed of small blocks with numerous pixels and large areas. In a specific environment, it can dissolve the decrease in image clarity caused by linear filters and efficiently eliminate pulse effects and scanning noise in the image. However, for some images with more details, especially points, lines, and high precision, using median filtering techniques is not suitable.

When in a one-dimensional state, the median filter is a window that includes an infinite number of pixels. After operation, we can use the pixel grayscale data at the center of the window and replace it with the average of all pixel grayscale data within the window. For example, if the length of the window is 5 and the grayscale values of its internal pixels are 80, 90, 200, 110, and 120, then this value is 110. Due to being arranged in ascending (or descending) order, the third digit value is 110, so the grayscale value of 200 originally located at the center of the window is replaced by 110. If 200 is the peak of noise, it will be removed. However, once this signal is filtered, its resolution will be reduced. Therefore, in some cases, median filtering can suppress noise, but in other cases, it can suppress the signal.

There is a one-dimensional sequence $f_1, f_2, \mathrm{L}\ , f_n$*. If the window length is an odd number m (where m is an odd number), median filtering is applied to this sequence by sequentially extracting m numbers from the input sequence,* $f_{i-v}, \mathrm{L}\ f_i, \mathrm{L}\ , f_{i+v}$*, Among them,* f_i *is the center value of the window,* $v=(m-1)/2$*, Arrange the values of point m according to their magnitude, and take the number in the middle as the filtering output. The expression for median filtering is:*

$$Y_i=Med\left\{f_{i-v}\mathrm{L}\ f_i\mathrm{L}\ f_{i+v}\right\}\qquad i\in Z\qquad v=(m-1)/2$$

Median filtering is a processing method for digital images, with its core being the median filtering of two-dimensional sequences. Although filtering windows are also two-dimensional, they can take on various shapes such as linear, square, circular, cross shaped, and undisturbed. The two-dimensional median filtering can be described as:

$$Y_{ij}=\sum_{A} Med\left\{X_{ij}\right\}$$

A is the window

Usually, we prioritize window 3, followed by window 5, and this process continues until we meet our expectations for the filtering results.

2.2.4. Image enhancement and smoothing in this article

In image enhancement processing, the core task is to adjust the grayscale of the image. The main objective of this study is to improve the clarity of contour lines. Due to the excellent performance and ease of calculation of histogram balancing technology in improving the clarity of contour lines, we have chosen to use this method for enhancement. The purpose of this article is to identify and classify car models, therefore there is a strict requirement for the clarity of object edges. Therefore, we have chosen to use median filtering technology to achieve image smoothing processing.

The specific procedure is as follows:

```
J=imnoise(I2,'salt & pepper',0.02);
K1=medfilt2(J,[3,3]);
```

```
imshow(K1);
imhist(K1);
[J,T]=histeq(K1,64);
Figure,imshow(J);
```

2.3. Edge detection

2.3.1. Overview of Edge Detection

Edges represent the brightest areas in an image, typically appearing at the boundaries between objects, objects, backgrounds, and various regions (including different colors). This is crucial for image segmentation, feature extraction, and other image research. Edge detection is usually the primary step in image analysis and understanding. Given the criticality of edge recognition, it has become one of the hottest topics in the field of machine vision. The focus of this section is to explore the core concepts of edge detection and use several commonly used edge detection devices to illustrate the key challenges of edge detection.

Before discussing edge computers, we need to clarify the meaning of the following concepts.

Edge point: a point in the image where the brightness changes significantly.

At the edge, we combine the coordinates of [i, j] with their directions, and the direction of this edge may be the gradient angle.

Edge recognition technology: a computational method that can extract boundaries (including boundary points or segments) from images.

Shape: A boundary graph, or the curve shape of a boundary graph.

Boundary linking: The process of converting an irregular boundary map into a regular boundary map. Usually, we sort the edges clockwise.

Boundary tracking is a search step aimed at identifying filtered contour images.

The coordinates of edge points can be represented by integer labels in rows and columns, or within the resolution range of sub pixels. Although edge coordinates may appear in the coordinate system of the initial image, most of the time they are mapped to the coordinate system of the output image generated by the edge detection filter, as the filtering operation may cause the coordinates of the image to shift or compress. The edge can be described by small line segments with pixel size, or by a point with directional features. Please note that in real usage scenarios, edge points and edge parts are uniformly defined as edges.

The main difference between edge connection and edge tracking is that the former takes the chaotic edge data created by the edge detector as input, and then obtains an organized edge data. The process of edge tracking is to generate a structured set of edge data based on a single photo. In addition, boundary recognition utilizes regional data to confirm boundaries, while boundary tracking relies on panoramic data to determine whether a pixel is a boundary.

2.3.2. Edge detection gradient operator

The gradient edge detection algorithm has the following four steps:

1) Filtering: The core of edge reconnaissance technology lies in the first and second derivatives, however, the operation of these derivatives is very sensitive to noise, so we need to use filtering devices to improve the performance of edge reconnaissance equipment related to noise. It is pointed out that most filters also cause a decrease in edge strength during the process of reducing noise, therefore, a balance needs to be achieved between improving edge quality and reducing noise.

2) Enhancement: The fundamental way to enhance boundaries is to set the intensity change values for different regions. By strengthening the algorithm, we can clearly see the parts that have undergone significant changes in the field. Usually, edge enhancement is achieved by calculating the gradient amplitude.

3) Recognition: Many points in the image have relatively high gradient amplitudes, but not all of these points are located at the edges in specific application scenarios. Therefore, we need to adopt a strategy to identify which points belong to edge points. Gradient, amplitude, and threshold are the most fundamental evaluation criteria for boundary detection.

4) Localization: Assuming an application needs to clarify the position of the edge, the position of the edge can be predicted through sub-pixel resolution, and the direction of the edge can also be predicted.

In edge detection schemes, the first three steps are widely used. Now, we have developed many edge detectors, and here we will explore several common edge detectors used for gradient computation.

1. Roberts operator

The Roberts crossover operator provides a simple approximation algorithm for gradient amplitude calculation:

$$G(i,j)=\left|f(i,j)-f(i+1,j+1)\right|+\left|f(i+1,j)-f(i,j+1)\right|$$

Using the convolution template representation method, the above equation becomes:

$$G(i,j)=\left|G_x\right|+\left|G_y\right|$$

Where G_x (on the left) and G_y (on the right) are calculated using the following template:

1	*0*		*0*	*-1*
0	*-1*		*1*	*0*

It is very important to calculate the true partial derivatives at the same position (x, y) in space when calculating gradients. The gradient approximation values G_x and G_y calculated using the 2x2 domain template mentioned above are not located at the same position. In fact, G_x is the approximate gradient at inner margin $(i, j+1/2)$, and G_y is actually the approximate gradient at inner margin $(i+1/2, j)$. Therefore, the Roberts operator is an approximation of the continuous gradient, rather than an approximation at the expected point (i, j). So, gradient values are usually calculated using the 3x3 domain.

2. Sobel operator

As mentioned earlier, using the 3x3 domain can avoid calculating gradients on interpolation points between pixels. Consider the arrangement of points around point (i, j) shown in the following figure.

a_0	a_1	a_2
a_7	(i,j)	a_3
a_6	a_5	a_4

The Sobel operator is also a type of gradient amplitude:

$$M = \sqrt{S_x^{\,2} + S_y^{\,2}}$$

The partial derivative is calculated using the following formula:

$$S_x = (a_2 + ca_3 + a_4) - (a_0 + ca_7 + a_6)$$
$$S_y = (a_6 + ca_5 + a_4) - (a_0 + ca_1 + a_2)$$

The constant coefficient c is 2.

Like other gradient operators, S_x *and* S_y *can be represented by convolution templates as follows:*

−1	0	1
−2	0	2
−1	0	1

1	2	1
0	0	0
−1	−2	−1

This calculation method focuses on the image elements near the core of the model. The Sobel operator is widely used in the field of edge detectors.

3. Prewitt operator

The equations of the Prewitt operator and Sobel operator are exactly the same, except that the constant coefficient c=1. So S_x *and* S_y *can be represented by convolution templates as follows:*

-1	0	1
−1	0	1
−1	0	−1

1	1	1
0	0	0
−1	−1	−1

2.3.3. Laplacian operator

In the previous discussion, we have explored edge detectors for first-order derivatives. If the first derivative we obtain exceeds a specific threshold, we can determine that this point is an edge point. This may lead to excessive boundary points. One optimization strategy is to use second-order derivative calculation. The Laplacian operator is widely used in second-order derivative operators.

The second-order smooth step edge derivative is a function that crosses zero at the edge point. The second-order equivalent of the second derivative is called the Laplacian operator factor, and its formula is:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Use the difference equation to approximate the second-order partial derivatives in the x and y directions as follows:

$$\frac{\partial^2 f}{\partial x^2} = \frac{\partial G_x}{\partial x} = \frac{\partial (f(i, j+1) - f(i, j))}{\partial x}$$
$$= \frac{\partial f(i, j+1)}{\partial x} - \frac{\partial f(i, j)}{\partial x} = f(i, j+2) - 2f(i, j+1) + f(i, j)$$

The approximate formula above is centered around point (i, j+1), and the approximate formula centered around point (i, j) is:

$$\frac{\partial^2 f}{\partial x^2} = f(i, j+1) - 2f(i, j) + f(i, j-1)$$

Similar ones include:

$$\frac{\partial^2 f}{\partial y^2} = f(i+1, j) - 2f(i, j) + f(i-1, j)$$

Integrate the previous two formulas into one calculation formula, and then describe it using an approximate Laplace calculation formula:

$$\nabla^2 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Once the output of the Laplacian operator reaches zero, it indicates the existence of edges, but it is necessary to remove those zero points that have no practical significance (i.e. areas with zero grayscale values).

2.3.4. LoG (Laplacian Gauss) operator

The essence of gradient operators and Laplacian operators is actually differential or integral algorithms, so these algorithms are very sensitive to noise. Therefore, before conducting boundary exploration, we need to eliminate noise. We constructed the LoG algorithm by integrating Gaussian filtering and Laplacian edge detection. The main features of LoG edge detector include:

(1) The smoothing filter is a Gaussian filter;

(2) In the enhancement process, we use the second derivative (two-dimensional Laplace function).

(3) Based on the intersection of the second derivative zero and the highest peak of the first derivative, we can perform edge recognition.

(4) Using linear interpolation techniques to predict the position of edges at sub-pixel resolution.

The advantage of this strategy is that it allows the image to initially interact with a Gaussian filter, which not only makes the image smoother but also reduces noise, and eliminates independent noise sources and smaller structural parts. Considering that smoothness may cause boundary expansion, we only consider regions with local gradient peaks as boundary areas during boundary exploration. The zero crossing point of the second derivative can achieve this goal. Due to Laplace being an undirected operator, it is used to approximate two-dimensional second-order derivatives. To prevent the discovery of ambiguous boundaries, we need to select zero intersections where the first derivative exceeds a specific threshold as boundary markers.

The LoG computer performs boundary exploration on images, and the results are generated by convolution operations, that is:

$$h(x, y) = \left[\left(\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} \right) e^{\frac{x^2+y^2}{2\sigma^2}} \right] * f(x, y)$$

If the image is too smooth, it may cause the boundaries to become unclear. Gaussian smoothing operation can blur the edges and other sharp and incoherent areas of the image, and the degree of blurring is determined by the value of the data. Although the higher the value, the better the noise filtering effect, it also means that the importance of edge information is lost, which affects the performance of edge detectors. If a smaller value is chosen, excessive noise may be generated due to insufficient smoothing. When a high-value filter smooths two adjacent edges, they may be connected together, so only one edge can be detected. So, if we don't know the size and position of the object, it's very difficult to determine the value of the filter. Usually, larger filters are used to generate stable edges, while smaller filters can produce precise localization edges. By combining these two, the optimal edge of the image can be identified.

2.3.5. Canny operator

A standard boundary symbolizes a change in the intensity of a jump, which is a characteristic of Gaussian noise. According to this model, an excellent boundary detector needs to meet three key elements: ① reducing the probability of error occurrence, which means minimizing the loss of actual boundaries while also preventing irrelevant nodes from being treated as boundaries to the greatest extent possible. ② High precision means we need to check as close as possible to the real boundaries. ③ Only unique feedback can be generated for each edge point to obtain a single pixel width boundary. The Cannae equation provides three criteria for boundary exploration equations.

(1) Signal-to-noise ratio criterion

The higher the signal-to-noise ratio, the higher the quality of the extracted edges. The signal-to-noise ratio (SNR) is defined as:

$$SNR = \frac{\left| \int_{-w}^{+w} G(-x)h(x)dx \right|}{\sigma \sqrt{\int_{-w}^{+w} h^2(x)dx}}$$

Where G (x) represents the edge function, h (x) represents the impulse response of a filter with a width of W, and represents the mean square error of Gaussian noise.

(2) Positioning accuracy criteria

The edge positioning accuracy is defined as follows:

$$L = \frac{\left| \int_{-w}^{+w} G'(-x)h'(x)dx \right|}{\sigma \sqrt{\int_{-w}^{+w} h'^2(x)dx}}$$

Among them, $G'(x)$ and $h'(x)$ are the derivatives of G (x) and h (x), respectively. The larger the L, the higher the positioning accuracy.

(3) Single edge response criterion

To ensure that each edge produces only one reaction, it should be ensured that the average distance D (f) of the zero crossing point of the impulse response derivative meets the following conditions:

$$D(f)=\pi\left\{\frac{\int_{-w}^{+w}h'^{2}(x)dx}{\int_{-w}^{+w}h''^{2}(x)dx}\right\}^{\frac{1}{2}}$$

$h''(x)$ is the second derivative of h (x).

Based on the aforementioned standards and rules, by adopting the strategy of functional differentiation, we can infer that the Canny edge detector is an optimal approximation operator, which behaves similarly to the first derivative of a Gaussian function. By integrating Kanni's three principles, we can obtain the optimal detection operator. The following are the steps of the Kanni edge detection algorithm:

(1) Smooth the image using a Gaussian filter;

(2) Use the first-order partial derivative finite difference method to estimate the magnitude and direction of gradients.

(3) Non maximum suppression of gradient amplitude;

(4) Use the dual threshold method to detect and connect edges.

2.3.6. Implementation of Edge Detection in Matlab

All previous operators can perform edge detection using the edge function in the Matlab image processing toolbox. The edge function contains numerous models and can set some models to have special reactions to horizontal or vertical boundaries (or both). The edge function can set a grayscale threshold, and only points that meet this threshold are recognized as boundary points. The main usage methods of the edge function are as follows:

$$BW=edge(I,'type',parameter,\mathrm{L})$$

Among them, I represents the input image, type represents the type of algorithm used, and parameter is a parameter related to the specific operator.

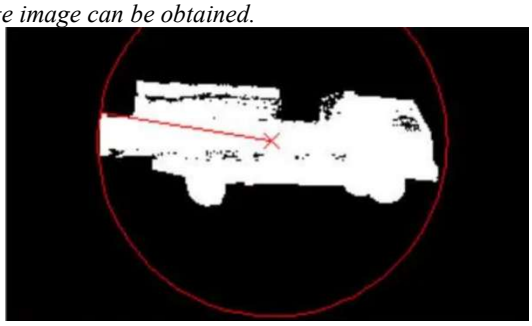
In this article, the Sobel operator is directly used for detection, and the program is as follows:

```
BW=edge(J,'sobel');
imshow(J);
figure,imshow(BW);
```

Further process the obtained edge image as follows:

```
bw=bwperim(BW);        %Refinement (Skeleton)
bw = bwareaopen(bw, 10);        %Remove small objects with pixels less than 10
bw=imclearborder(bw,4);        %Clear objects connected to the boundary
figure,imshow(bw);
```

Ultimately, a complete and clear edge image can be obtained.



3. Feature selection of vehicle image

3.1. Problem Proposal

Image recognition relies on certain specific image attributes, and obviously, the selection of these attributes is crucial, which has a profound impact on the construction, performance, and accuracy of image recognition analyzers. If there are problems with the selection of features, our classification will lose accuracy, and sometimes it may even be completely impossible to proceed. So, in image recognition, feature selection is an important step. Because in practical problems, it is often difficult to find the most critical features, or certain image features may change with changes in the environment, which makes feature selection and extraction complex.

The main goal of selecting and extracting features is to find the most effective feature among numerous features. When the total sample size is small, it is not reasonable to use a large number of features to construct a classifier, whether from the perspective of computational difficulty or classifier performance. So, selecting and refining features is a crucial and difficult step.

3.2. Some Basic Concepts

1. Feature formation

Based on the image to be identified, a series of initial features are generated through computation, which is called feature construction'

2. Feature extraction

When the scale of primary features is large, that is to say, the sample of the image is located in a higher dimension, we can use mapping techniques to present the sample in a smaller dimension, and this operation is called feature extraction. After mapping, the obtained feature is a certain type of the original feature. From a broader perspective, feature extraction is actually a form of transformation.

3. Feature selection

Among a series of features, selecting some of the most effective features to achieve the goal of reducing spatial dimensions is called feature filtering. At present, there is no clear strategy to guide the selection of features. Most of the time, relying on intuition, we can develop a list of possible features and then use feature ranking techniques to estimate the recognition rate of various features. By reducing the results, we can select the best features.

Good features should have the following four characteristics:

(1) Characterized by discernment. There must be distinct differences in the characteristics of various types of images.

(2) Reliability. In similar images, their characteristic values should be as similar as possible.

(3) Strong autonomy. The selected attributes are not directly related.

(4) The number is relatively small. With the rapid increase in system dimensionality (number of features), the complexity of image recognition systems is also constantly increasing. The importance of training classifiers and obtaining test results from image samples lies in the exponential increase in the number of features. In addition, adding noisy features or features that are highly correlated with existing features can actually lead to a decrease in the performance of the recognition system.

In practical use, feature extraction is usually carried out by first verifying a set of visually appropriate features, and then reducing them to a certain number to form a set that meets the requirements. Generally speaking, there are almost no optimal properties that meet the above criteria.

3.3. Feature extraction in this paper

At present, there are many types of cars, and how to choose appropriate features to identify and classify vehicle models is a challenge and key. The effectiveness of obtaining characteristics directly affects the efficiency and accuracy of recognition and classification. Generally speaking, side views of various vehicle models can provide relevant data such as the overall length of the body, the position of the top, the length of the top, and the height of the vehicle. Because in the process of image capture, the subtle distance and angle conversion between the lens and the vehicle may result in differences in the absolute values of all parameters for the same vehicle in two captured images, which makes it impossible for us to use absolute values such as absolute length or absolute height as recognition markers. According to the nature of use of automobiles, we can divide them into three types: passenger cars, sedans, and trucks. After conducting data research on three categories of vehicles, the standard we selected is the sum of the lengths of the car's head and tail, which is defined as the head length ratio. The aspect ratio, also known as the top to top ratio, refers to the difference between the length of a car's top and its height. Taking the centerline of the canopy as the boundary and comparing it to other areas is called the front to back ratio.

Therefore, we need to measure the total length, total height, and top length of the car.

The first thing is to determine the total length of the car, and the detailed calculation method is as follows:

In a complete image, we scanned the entire binary image from left to right, searching for pixels with a grayscale value of 0. Once we find a pixel A, we can obtain its horizontal axis Xa. Scan from right to left again to find a pixel with a grayscale value of 0. Once a pixel B is discovered, its horizontal axis Xb can be obtained. At this point, by subtracting the horizontal coordinates of two points, we can calculate the length of the car (Length=Xb Xa).

Next, calculate the overall height of the car, and the specific algorithm is as follows:

For a complete two-dimensional image, we need to detect each pixel from top to bottom, looking for those pixels with a gray level of 0. Once we discover a pixel A, we can obtain its horizontal axis Ya. We then perform detection from right to left, searching for images with grayscale values of 0. Once we discover an image B, we can obtain its abscissa Yb. At this point, by subtracting the abscissa of two points, we can calculate the height of the car (Height=Yb Ya).

Finally, the calculation of the car roof is as follows:

In the visual interval from Yb to Yb+n (where n represents empirical data), we scan the image from left to right to find pixels with a gray level of 0. Once we discover a pixel A, we can obtain its horizontal axis Ta. We then perform detection from right to left, searching for images with grayscale values of 0. Once we discover an image B, we can obtain its horizontal coordinate Tb. At this point, by subtracting the horizontal coordinates of two points, we can calculate the length of the car roof (Top=Tb Ta).

After obtaining the three necessary data, we can calculate the three key parameters set in the experiment.

Top to length ratio=roof length/vehicle length (x1=Top/Length);

Top to Height Ratio=Roof Length/Vehicle Height (x2=Top/Height);

Front to rear ratio=front part/rear part {x3=(Ta+Tb)/2-Xa)/(Xb - (Ta+Tb)/2)};

The program for feature extraction is as follows:

```
[x,y]=size(bw); %Calculate the size of the entire image
for i=1:x
he=0;
for j=1:y
if (bw(i,j)==1)
he=he+1;
end
H(i)=he;
end
if H(i)>10
Ya=i;
```

```
break;
end
end
for i=x:-1:1
he=0;
for j=1:y
if (bw(i,j)==1)
he=he+1;
end
H(i)=he;
end
if H(i)>10
Yb=i;
break;
end
end
Height=Yb-Ya;          %Calculate the height of the car
for j=1:y
he=0;
for i=1:x
if (bw(i,j)==1)
he=he+1;
end
H(j)=he;
end
if H(j)>5
Xa=j;
break;
end
end
for j=y:-1:1;
he=0;
for i=1:x
if (bw(i,j)==1)
he=he+1;
end
H(j)=he;
end
if H(j)>5
Xb=j;
break;
end
end
Length=Xb-Xa;          %Calculate the length of a car
for j=1:y
he=0;
for i=Ya:Ya+8
if (bw(i,j)==1)
he=he+1;
end
T(j)=he;
end
if T(j)>=1
Ta=j;
break;
end
end
for j=y:-1:1;
he=0;
for i=Ya:Ya+5
if (bw(i,j)==1)
he=he+1;
end
T(j)=he;
end
if T(j)>=1
```

```
Tb=j;
break;
end
end
Top=Tb-Ta;        %Calculate the length of the car's roof
x1=Top/Length;        %Top length ratio
x2=Top/Height;        %Top height ratio
x3=((Ta+Tb)/2-Xa)/(Xb-(Ta+Tb)/2);        %Before and after comparison
X=[x1;x2;x3]
```

Collect 4 samples each for buses and sedans, with 3 samples for each type of vehicle used for training and 1 sample for recognition. The specific data are shown in Tables 1 and 2 below:

Table 1 Training Set Sample Data

model	Numbe rmode	Vehicle characteristic vector (Sample input value)		
		x1	x2	x3
bus	1	0.9268	3.0811	1.0500
	2	0.7087	2.7527	1.1597
	3	0.7000	2.0152	1.0207
sedan	4	0.3563	1.3367	1.3705
	5	0.3872	1.3582	1.4406
	6	0.3602	1.1972	1.4456

Table 2 Samples to be Identified

model	serial number	x1	x2	x3
bus	7	0.9639	2.8571	1.1558
sedan	8	0.3593	1.0000	1.4444

4. Design of BP Neural Network Classifier

4.1. Overview of Artificial Neural Networks

ANN (Artificial Neural Network) is a method of simulating the construction and functioning of human brain neural networks through engineering techniques, with the main goal of enabling machines to have perception, learning, and inference abilities similar to those of the human brain. Artificial neural networks can only achieve significant progress due to their following characteristics.

(1) The characteristic of neural networks is decentralized. By utilizing the connections of numerous neurons and assigning weights to each connection, neural networks can display specific data, so even if the network is damaged in certain regions, it can maintain the integrity of the original data.

(2) Neural networks exhibit synchronicity when processing data. All neurons can autonomously calculate and manipulate based on the acquired data, and then send the conclusions to others, which is a perspective of synchronous processing.

(3) Neural networks exhibit independent structure and self-learning ability when processing data. In neural networks, the connection strength of each neuron is measured by the size of its weights, which can be preset or adjusted according to changes in the external environment. This process is called the learning stage of neurons.

4.1.1. Research History of Neural Network Pattern Recognition

In the late 1950s and early 1960s, the introduction of Rosenb1att's Perceptron and Widrow's Adaline provided us with a fundamental linear classifier. However, due to the lack of effective training methods for Perceptron in dealing with more complex high-order predicate problems, coupled with Minsky and Papert's influential book Perceptron published in 1969, this research fell into a slump. In 1986, Rumelhart and McClelland first introduced the Error Backpropagation Algorithm (BP), which completely transformed the current situation. This development further promotes the educational application of the MLPN model. Subsequently, based on the multi-layer perceptron network, some feedforward networks have been derived, such as radial basis function network (RBFN), function chain network (FIN), etc.

In 1982, physicist Hopfield from the California Institute of Technology in the United States pioneered a feedback network model, which is mainly used for associative memory and optimization calculations. He introduced the concept of "energy function" on this basis, ensuring that artificial neural networks can effectively operate in specific circuits, which greatly promoted the progress of neural networks and reignited public enthusiasm for neural networks. In 1973, Watanabe further proposed the principle of utilizing pattern subspaces to

design subspaces for various categories. Through various subsets of classification, we are able to identify models, which is undoubtedly the most clear theory of model identification. In the late 1970s, Kohonen first proposed the theory of learning subspace (LSM) model classification. Oja further proposed the Average Learning Subspace Method (ALSM), which provides a relatively complete framework for subspace pattern recognition and has been successfully applied in experiments of speech phoneme recognition.

In fact, learning subspace technology is a neural network model recognition technique that does not require teacher guidance. In 1991, researchers such as L. Xu and E.0ja successfully applied numerous parallel architecture neurons to this technology, further connecting learning subspace technology with neural networks with hardware architecture, promoting the practical application of this field.

In 1962, Widrow first proposed the concept of adaptive linear element (Adaline), which was specifically designed for the research of adaptive learning systems in the brain at that time. In view of the fact that the exploration of neural networks was going to a low ebb in the late 1960s, Widrow mainly applied this model theory to the field of adaptive signal processing. In the latter half of the 1980s, scientists realized that these principles were actually the core rules of neural learning systems, and could also effectively handle adaptive pattern recognition problems.

In the 1970s, Anderson's BSB model, Kohonen's self-organizing feature mapping network model, and Grossberg et al.'s adaptive resonance theory (ART) provided deeper theoretical support for neural network pattern recognition theory. Undoubtedly, current neural network physics models have achieved significant results in pattern recognition. The growth of pattern recognition theory will be driven by some breakthroughs in neural network theory. On the contrary, in-depth exploration and improvement of pattern recognition theory will greatly promote the progress of neural network theory. Their connection is mutual interaction and influence.

4.1.2. General characteristics and basic models of neural networks

*According to the discoveries of neuroscience, the nervous system of life is mainly composed of numerous neural units and neurons tightly connected together. One neural unit is closely connected to another and communicates information through neural activity. The cortex of the human brain is constructed from six independent functional areas, covering approximately 1.4*neurons, which is equivalent to ten thousand large computers operating simultaneously. Therefore, we can say that the human brain is a broad and interconnected complex nonlinear system with unique information processing capabilities. The intention of exploring the human brain is to simulate the way humans process information, in order to build a new generation of intelligent computers. Therefore, in the field of engineering, our main focus is on artificial neural networks (ANN). Therefore, the basic explanation of artificial neural networks by American neural network expert Hecht Nielsen is that neural networks are computer systems composed of multiple simple processing units connected to each other in some way, and this system relies on its dynamic response to external input information to process information. So, the neural network we usually refer to actually refers to artificial neural networks. Artificial neural networks, based on simulating the human brain, inherit the main characteristics of the human brain, including:*

(1) The high parallelism of the human brain

The speed of information exchange (i.e. neural pulses) between neurons is defined as milliseconds, but in conventional computers, this speed may only be a few tens of microseconds. However, people are usually able to make accurate judgments about things in a very short amount of time. For example, when dealing with a general "multiple choice one" problem, the human brain can immediately make a large number of choices, while using a computer for selection requires a lot of search time. This indicates that the human brain must be built on the basis of large-scale parallel processing, reflecting that the human brain has different computing principles.

(2) The highly nonlinear global effects of the human brain

The neural network structure mainly consists of numerous basic neurons, which can obtain information from numerous other neurons and use the nonlinear transmission and feedback of this information to create new information, thereby affecting other neurons. The operation of a network involves mutual constraints and influences, achieving a nonlinear transformation from the input state space to the output state space. The development of the internet follows the law of global influence, gradually evolving from an input state to a final state and producing outputs. From a broad perspective, the overall effect of a network is not just a simplification of the effects of its various parts, but reflects the characteristics of a group. According to the serialized regional operation specifications, each step of the computer is closely related to the previous step and may also have an impact on the subsequent steps. These problems are gradually solved by algorithms.

(3) The good "fault tolerance" and associative memory function of the human brain

The human brain has the ability to quickly recognize friends who have not seen each other for many years and have undergone significant changes in their appearance. Being able to identify blurry or damaged photos and determine their true appearance indicates that our brain has a high level of "fault tolerance" and associative memory ability. In addition, although there is a large amount of cell death every day, it has not affected our brain function. In the field of computers, there are significant differences in the situation, and any damage to a component or minor error in a program can lead to serious impacts, demonstrating its enormous vulnerability.

(4) The human brain and computer have different ways of storing and processing information

Due to the significant differences in information processing methods between the human brain and computers, this is their main difference. In the human brain, many connection keys are used to express and remember knowledge and information, and these connection keys can also simultaneously store various types of information. In this way, the processing and storage of information become integrated, enabling various types of information to communicate naturally. Computers do not have a direct connection when storing various types of data and information, which means they are independent. Only by relying on manually written programs can they communicate, but this communication cannot exceed the expectations of the programmer. Due to this difference, we can see that computers can only make basic logical deductions, but the human brain has the ability to engage in deeper visual thinking, summarize, compare, and expand based on the information already mastered, quickly understand the overall situation, and make correct judgments and decisions.

(5) The adaptive and self-learning ability of the human brain

Although the human brain is largely limited by genetic factors, postnatal factors such as experience, exercise, and learning also play a crucial role. Many intellectual operations are not based on logical inference, but are shaped through habitual nature. For example, when night falls, we can quickly recognize the sounds of people we are familiar with. Children can quickly identify family members, but their detailed attributes are unknown. Without a clear calculation method, creating a program will be very difficult, which is vastly

different from computer-led logical reasoning. This indicates that our brain has extremely high characteristics of self adjustment and self-learning.

Usually, the layout of neural networks can be divided into two categories: forward propagating grids and backward propagating grids. Generally speaking, feedforward networks can be divided into MADLINES networks, MLPN multilayer perceptron networks, RBFN radial basis function networks, RBFN function chain networks, PNN probabilistic neural networks, and PNN high-order networks. Hopfield, Boltzmann, and ART feedback networks are generally widely used, however, Hamming networks are actually a collection of these two types of networks.

The four main ways of training and learning neural networks are one of them.

(1) Having guidance from an external supervisor: The characteristics of the models and types that need to be classified have been determined. In the input of each model sample, the output part of the network will generate a relevant guidance (monitoring) information to match its features. According to a certain objective function law, we can continuously adjust the monitoring signal and real output at the network output, thereby reducing the error rate at the network output to the set standard.

(2) Learning without a mentor (also known as self-organization): The characteristics of the pattern categories to be classified have not yet been determined. According to specific clustering rules, the construction and connection weights of the network will automatically learn and adjust the model samples of the surrounding environment, so that the distribution of network construction and connections can accurately display the statistical distribution of training samples.

(3) Self monitoring learning: Knowing the category attributes of the pattern to be classified. In each input of the model sample, the network can continuously optimize the connection distribution of the network based on the unique properties of the sample, following some rules, until the preset accuracy standards are met.

(4) Blended learning mode: a learning approach with and without mentors. The pattern and category attributes to be classified have been determined. We use unsupervised clustering to study the internal structure and parameters of the network, while also retaining two related guidance (or monitoring) signals at the output of the network. According to specific rules, by continuously adjusting the connection weights between the hidden layer and the output layer, the error between the network output and the monitoring signal is gradually reduced to a preset standard.

Among the four neural network training methods provided, it can actually be distinguished into two types: labeled pattern classification and unlabeled pattern classification based on whether the pattern samples have attribute labels. For unlabeled pattern classification, due to the unclear properties of the pattern, unsupervised clustering is usually chosen to optimize the encoding of pattern samples with unknown properties. However, for pattern classification problems with labels, in most cases, a supervision signal is set at the network output to force the input pattern to approach the expected supervision signal through the nonlinear mapping of each layer of the network. However, some classification networks do not accept external monitoring (mentor) signals, but directly use given labeling patterns to design or train network connection weights. Therefore, this learning method is called unsupervised learning without a mentor. In fact, in practical operation, unsupervised learning or self supervised learning networks often require manual selection of control parameters. So, in a broad sense, all learning methods are supervised learning, it's just that their specific forms are different.

It is worth emphasizing that the various neural network models currently proposed are only based on our research results on advanced animal brains and provide some insights. They are only one type of model in the field of intelligent machine engineering. From a rigorous perspective, these cannot yet reveal the operating principles of advanced biological brains. In the past few years, research in biology and physiology has mainly focused on the molecular level. However, our understanding of the entire structure of the human brain and its information processing methods is still quite limited.

The characteristics of using neural networks for image recognition include: the information is widely stored in the connection weight coefficients, which can provide good error correction ability. However, due to the frequent interference of noise or loss of input images in the image recognition process, neural networks can effectively handle these issues. In addition, due to the self construction and learning characteristics of neural networks, they can effectively alleviate the limitations of image recognition technology, thus demonstrating significant advantages in solving image recognition problems.

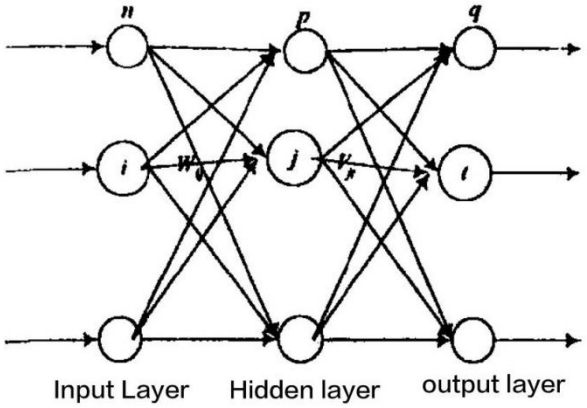
Due to its characteristics of dispersed information storage, synchronous operation, and self-learning, artificial neural network systems have been widely applied in various scenarios including information processing, model recognition, intelligent operation, and system construction. Especially the BP neural network, which is a multi-layer feedforward network based on the error backpropagation algorithm, can approximate any continuous function with any accuracy. Therefore, it has been widely used in fields such as nonlinear modeling, function approximation, and pattern classification. However, when using neural networks to handle practical problems, it inevitably involves a large number of numerical computation problems. Mathworks, a company based in the United States, has developed a highly powerful Matlab for handling conflicts between digital computing and computer simulation. Here, we use Matlab language to create a tool function based on neural network theory, which is what we call the Neural Network Toolbox. Matlab has specially designed numerous tool functions for BP neural networks, which are important supports for the practical application of BP neural networks.

4.2. Implementation of BP Neural Network Algorithm

The introduction of BP algorithm is a key factor in the success of BP network. The BP algorithm is a delta algorithm, which is a monitoring based learning method. The core idea of this theory is: M input learning data and their related output data. The purpose of learning is to use the difference between the real output of the network and the preset vector to adjust their weights, in order to match the current situation and expectations as much as possible. Even if the sum of squared differences at the output level of the network has been minimized, he will continue to measure the changes in network weights and differences in the direction of decreasing the inclination of the corresponding error function, gradually approaching the preset target. All changes in weights and errors will be affected by network errors, which will spread to various levels through reverse transmission.

The BP algorithm consists of two main parts: forward transmission of information and backward diffusion of errors. In the process of feedforward processing, information starts from the input layer in sequence, goes through multiple hidden layers for step-by-step operations, and finally converges to the output layer. At the same time, the data produced by the neural units at each level will become the influencing factor for the input of the neural units in that layer. If the end layer fails to produce the expected result, the error difference of that layer is calculated, and then the reverse propagation process is performed to backpropagate the error information on the original path, in order to adjust the weight parameters of each layer of the network until the desired goal is achieved.

Using the three-layer network structure shown in the diagram as an example, explain the execution steps of the BP network algorithm.



1. Forward propagation of input mode

The activation values of each neuron in the hidden layer are:

$$S_j = \sum_{i=1}^{n} W_{ij} a_i - \theta_j \qquad j = 1,2,\mathrm{L}\ , p$$

The output values of each neuron in the hidden layer are:

$$b_j = f(s_j) \qquad \begin{matrix} V_{jt}(t+1) = V_{jt}(t) + \alpha \delta_t b_j \\ W_{ij}(t+1) = W_{ij}(t) + \beta \delta_j b_i \end{matrix} \qquad j = 1,2,\mathrm{L}\ , p$$

Similarly, the activation values of each neuron in the output layer are:

$$l_t = \sum_{j=1}^{p} V_{jt} b_j - \gamma_t$$

The actual output values of each neuron in the output layer are:

$$c_t = f(l_t)$$

2. Output error backpropagation

The deviation between the desired output of the network and the actual output is:

$$\delta_t = y_t - c_t$$

Calculate using the sum of squares error:

$$E = \frac{1}{2} \sum_{t=1}^{q} (y_t - c_t)^2 = \frac{1}{2} \sum_{t=1}^{q} \delta_t^2$$

Output error of each neuron in the output layer:

$$\delta_t = (y_t - c_t) f'(l_t)$$

Output error of each neuron in the hidden layer:

$$\delta_j = f'(s_j) \sum_{j=1}^{m} \delta_t V_{jt}$$

Correction weight:

$$V_{jt}(t+1) = V_{jt}(t) + \alpha \delta_t b_j$$

$$W_{ij}(t+1) = W_{ij}(t) + \beta \delta_j b_i$$

4.3. Design of BP Neural Network Classifier

Matlab's neural network library makes building neural network models simple and convenient. The following will use a three-layer neural network as an example to illustrate the construction method of a backpropagation network.

4.3.1. Training steps

(1) Initialize the weights w and threshold b of each layer with small random numbers to ensure that the network is not saturated by large weighted inputs, and also perform the following parameter initialization:

Set the minimum expected error value: err_goal;

Set maximum number of cycles: max epoch;

Set learning rate: usually choose lr=0.02-0.7;

(2) Calculate the output vectors A1 and A2 of each layer of the network, as well as the network error E, and the corresponding output samples are T:

*A1 = logsig (w1*p ,b1) ;*

*A2 = purelin (w2 *A1 ,b2) ;*

E = T-A2 ;

(3) Calculate the error changes D2 and D1 of each layer's backpropagation, and calculate the correction values of each layer's weights as well as the new weights:

D2 = deltalin (A2 ,E) ;
D1 = deltalog (A1 ,D2 ,w2) ;
[dw1 db1] = learnbp (p ,D1 ,lr) ;
[dw2 db2] = learnbp (A1 ,D2 ,lr) ;
w1 = w1 + dw1 ; b1 = b1 + db1 ;
w2 = w2 + dw2 ; b2 = b2 + db2 ;
(4) Calculate the sum of squared errors after correction again:
*SSE = sumsqr (T-purelin (w2*logsig (w1*p ,b1) ,b2)) ;*
(5) Check if SSE is less than err_goal, if so, the training ends; Otherwise, continue.

4.3.2. Design of BP Neural Network

(1) Initialize forward network

Initialization is the process of initializing the connection weights and thresholds. The initff function automatically calls the initialization function while establishing network objects, initializing the network's connection weights and thresholds based on default parameters.

Format:
[w1 ,b1 ,w2 ,b2] = initff (p ,s1 ,f1 ,s2 ,f2)

Among them, p represents the input vector, s represents the number of neurons, f represents the transfer function, w represents the weight, and b represents the threshold.

(2) Training Network

After initialization, the BP network can be trained. The function uses batch processing to update the network connection weights and thresholds, and its parameters need to be set, such as learning step size, error target, etc. At the same time, during the network training process, the network error is also displayed graphically as a function of the number of learning iterations.

① Basic gradient descent method for training network function trainbp
Format:
[w1 ,b1 ,w2 ,b2 ,te ,t r] = t rainbp (w1 ,b1 ,f1 ,w2 ,b2 ,f2 ,p ,t ,tp)
② Training network function trainbpx using adaptive learning algorithm with momentum term
Format:
[w1 ,b1 ,w2 , b2 , te , t r] = t rainbpx (w1 , b1 ,f1 ,w2 , b2 ,f2 ,p ,t ,tp)

Among them, p represents the input vector, t represents the target vector, te is the actual number of training iterations of the network, tr is the row vector of the sum of squares of the network training errors, and tp represents the network training parameters (such as learning rate, expected error, maximum learning iterations, etc.).

(3) Network simulation

The simulation function simff is used to simulate the network. By using this function, input and output simulations can be conducted before and after network training to compare and modify the network for evaluation.

Format:
a = simff (p ,w1 ,b1 ,f1 ,w2 ,b2 ,f2)
Where a represents the actual output of the trained BP network.

4.3.3. Design of neural network classifier in this article

In this article, the training of BP network adopts the following procedure:

```
function car2
%BP neural network vehicle recognition testing system
clc
Fprintf ('★★★★★ BP neural network vehicle recognition testing system ★★★★ \ n ')
Fprintf ('★★★★★★---------- Press any key to start ---------- ★★★★ ')
pause
close all
clear
echo on
clc
% 1. Load training samples and target matrix
Fprintf ('Neural network initialization, press any key to start \ n ')
pause
clc
load car1
image=[X1,X2,X3,X4,X5,X6];
image=double(image);        %Type conversion
target=[1 0;1 0;1 0;0 1;0 1;0 1]';        %Target Matrix
[R1,Q1]=size(image);
[R2,Q2]=size(target);
clc
% 2. Generate neural network
S1=10;
S2=R2;
net=newff(minmax(image),[S1 S2],{'purelin' 'purelin'},'traingdx');        %Newff - Generate a new forward neural network
net.LW{2,1}=net.LW{2,1}*0.01;        %Current input layer weights and thresholds
net.b{2}=net.b{2}*0.01;
```

```
clc
% 3. Training neural network
Fprintf ('Start training neural network, press any key to continue \ n')
pause
clc
%3.1 Training Neural Networks with Ideal Samples
Fprintf ('Start training neural network with ideal samples, press any key to continue \ n')
P=image;
T=target;
net.performFcn='sse';
net.trainParam.goal=0.1;
net.trainParam.show=20;
net.trainParam.epochs=500;
net.trainParam.mc=0.95;
[net,tr]=train(net,P,T);          %Train - Train the BP neural network
pause
clc
%3.2 Training Neural Networks with Samples Containing Different Levels of Noise
Fprintf (Start training neural networks with samples containing varying degrees of noise, press any key to continue \ n ')
netn=net;
netn.trainParam.goal=0.1;
netn.trainParam.epochs=300;
T=[target target target target];
for pass=1:10
fprintf('Pass=%.0f\n',pass);
P=[image,image,(image+randn(R1,Q1)*0.1),(image+randn(R1,Q1)*0.2)];
[netn,tr]=train(netn,P,T);
echo off
end
echo on
Paul, press any key to continue
clc
%3.3 Training the network again with ideal samples
Fprintf ('Train the network again with ideal samples, press any key to continue \ n')
netn.trainParam.goal=0.1;
netn.trainParam.epochs=500;
netn.trainParam.show=5;
P=image;
T=target;
[netn,tr]=train(netn,P,T);
%3.4 Training End
pause
clc
% 4. Test the recognition network
Fprintf ('Test the recognition network, press any key to continue \ n')
pause
clc
%4.1 Setting Test Parameters
noise_range=0:0.1:1.0; %Mean square error of noise<--------- parameter ----------
max_test=100;
network1=[];         %The misidentification rate of Network 1
network2=[];         %The misidentification rate of Network 2
%4.2 Conduct network simulation and testing
for noiselevel=noise_range
fprintf('Testing networks with noise level of %.2f.\n',noiselevel);
errors1=0;
errors2=0;
for i=1:max_test
P=image+randn(3,6)*noiselevel;
%Test Network 1
A=sim(net,P);          %Sim - Simulate BP Neural Network
AA=compet(A);
errors1=errors1+sum(sum(abs(AA-T)))/2;
%Test Network 2
An=sim(netn,P);
AAn=compet(An);
```

```
errors2=errors2+sum(sum(abs(AAn-T)))/2;
echo off
end
network1=[network1 errors1/3/100];
network2=[network2 errors2/3/100];
end
echo on
Fprintf ('Draw result curve - misidentification rate curve, press any key to continue \ n')
pause
%4.3 Draw the result curve by typing any key - the misidentification rate curve
clc
%Draw a network misidentification rate curve
plot(noise_range,network1*100,'--',noise_range,network2*100);
title('Percentage of Recongnition Errors(%)');
xlabel('Noise Level');
ylabel('Network 1 - -  Network 2 ---');
% 5. Input features for recognition
Fprintf ('Input features for recognition, press any key to continue')
pause
clc
save car2.mat
```

After training the network, run the following program to recognize the vehicle model:

```
function rec(X)
load car2
input=X;
input=reshape(input,3,1); %
mid=sim(net,input);
mid=compet(input);
answer=find(compet(mid)==1);          %Answer refers to the matched columns in the sample set
switch answer
case  1|2|3
Fprintf ('Bus')
otherwise
Fprintf ('sedan ')
end
pause
clc
echo off
```

5. Analysis and Discussion of Experimental Results

5.1. Analysis and Discussion of Preprocessing Results

After capturing images of various types of cars, image preprocessing is necessary before feature recognition. The preprocessed image is shown below:



Figure 1



Figure 2



Figure 3



Figure 4

Among them, Figure 1 shows the original car model image captured, Figure 2 shows the binarized image, Figure 3 shows the image obtained after edge detection, and Figure 4 shows the image obtained after removing some isolated points after edge detection.

The final image obtained in this experiment has clear edges and few isolated points, which is beneficial for feature extraction. The image processing performed in this experiment includes histogram homogenization enhancement, median filter filtering, and Sobel operator edge detection. In the preprocessing process, processing can be carried out according to the specific situation of the obtained image. When enhancing images, Fourier transform enhancement can be used for enhancement; When filtering images, Gaussian filtering,

low-pass filtering, and other methods can be used for filtering; In edge detection, pre witt operator, Canny operator, etc. can be used for edge detection, or one can write their own algorithm for edge detection.

5.2. Analysis and Discussion of Feature Extraction Results

The results of feature extraction in this experiment are shown in Tables 1 and 2 in Chapter 3. The obtained eigenvalues basically meet the requirements. The top to length ratio of buses is close to 1, while that of sedans is around 0.3; The top to height ratio of buses is about 2.5, while that of sedans is about 1; The front to rear ratio of a passenger car is about 1, and the front to rear ratio of a sedan is about 1.5. By using the three characteristics of top to length ratio, top to height ratio, and front to rear ratio, it is possible to distinguish between passenger cars and sedans.

In this experiment, the projection method was used to extract vehicle features. By projecting the entire vehicle model image in the horizontal axis direction, the areas with dense pixel points on the left and right sides can be obtained, thereby determining the length of the car; Projection of the entire vehicle model image in the vertical axis direction can obtain the areas with dense pixel points on both the upper and lower sides, thereby determining the height of the car; Finally, the length of the car roof can be obtained by projecting the empirical value of the roof and below in the horizontal axis direction.

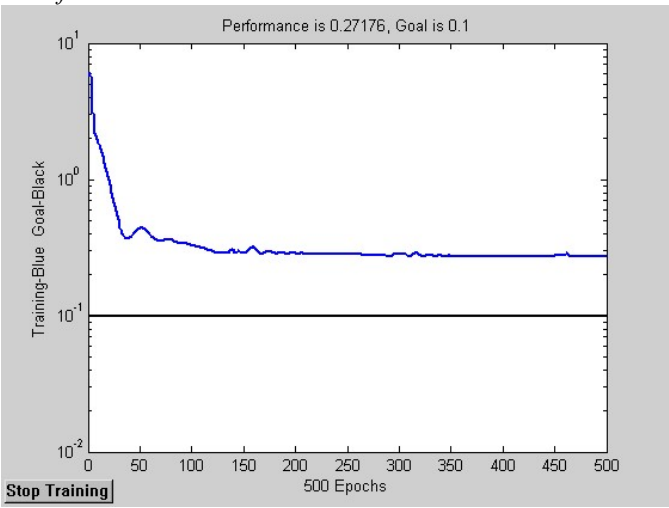
Before training and recognizing with a BP neural network, in order to facilitate the training of the BP network, a feature set needs to be created. The program is as follows:

```
function car1
X1=[0.9268;3.0811;1.0500];
X2=[0.7087;2.7527;1.1597];
X3=[0.7000;2.0152;1.0207];
X4=[0.3563;1.3367;1.3232];
X5=[0.3872;1.3582;1.2705];
X6=[0.3602;1.1972;1.4456];
X7=[0.9639;2.8571;1.1558];
X8=[0.3593;1.0000;1.4444];
save car1.mat
```

To achieve automatic recognition of car models, additional features can also be extracted. If a certain method is used to detect the number of axles of a car, the detection results will be used as a standard for classifying the car; Analyze and search for vehicle model information from various information on license plates; Based on the positioning of the car edge and windows to achieve vehicle recognition, etc.

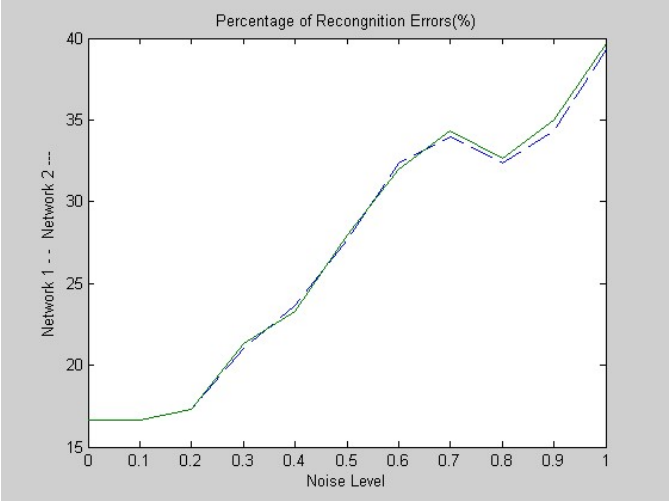
5.3. Analysis and Discussion of BP Neural Network Recognition Results for Vehicle Models

After feature extraction of the vehicle model, the BP neural network was trained using 6 samples (3 for buses and 3 for sedans). The curve graph of its training function is as follows:



From the above curve, it can be seen that the decline is relatively fast in the early stage of training, resulting in a large error in network testing.

After simulating and testing the network, the network misidentification rate curve obtained is as follows:



From the network misidentification rate curve above, it can be seen that the misidentification rate continues to increase as the noise gradually increases. This indicates that there are fewer samples used for training, which can easily lead to misidentification. To improve the accuracy of vehicle recognition, more samples need to be selected for training.

Finally, the vehicle model is identified, that is, samples 7 and 8 are identified. Enter the "rec (X7)" command in the Matlab command line, and the words "bus" will be displayed; When entering the "rec (X8)" command in the Matlab command line, the words "sedan" will be displayed.

6. Full text summary

This paper is based on the public's classification standards for vehicles, capturing various types of car photos and extensively preprocessing the collected images to extract necessary characteristic data. Using neural networks, preliminary identification of vehicle models is achieved, including image preprocessing, feature extraction, and BP neural network classifier construction.

In this article, our first step is to perform preliminary processing on the image to generate a thoroughly binarized edge map, laying the foundation for subsequent feature extraction work. Next, three parameters were extracted: crown length ratio, crown height ratio, and aspect ratio. The ultimate step is to build a backpropagation neural network to implement the identification of vehicle types.

This paper mainly explores the process of extracting vehicle model attribute data and constructing a BP neural network classification model, with the research focus on the extraction of characteristic data.

In summary, this study provides an innovative solution for autonomous identification of vehicle types, provides a basis for the implementation of automatic transportation toll collection systems, provides evidence for post verification of manual payments, and enhances the statistical database of the transportation management bureau, helping to make traffic management automation a reality as soon as possible.

References

1. Yuan, Z. (2019). *Artificial neural networks and applications*. Tsinghua University Press.
2. Rong, G. (2022). *Computer image processing*. Tsinghua University Press.
3. Wang, N., Ren, B., & others. (2021). Automatic recognition of car model images based on neural networks. *Chinese Journal of Image and Graphics*, (8), 668–672.
4. Lou, S., & Shi, Y. (2023). *System analysis and design based on Matlab – Neural networks*. Xi'an University of Electronic Science and Technology Press.
5. Wang, S. (2023). *Matlab 6.5 assists in image processing*. Electronic Industry Press.
6. Yin, G. (2022). Research on the application of fuzzy neural networks in automatic vehicle recognition. *China Mechanical Engineering*, *13*(2), January.
7. Liu, H., & Bai, Z. (2020). Research on Matlab implementation and application of BP network. *Modern Electronic Technology*, *2*(217).
8. Zhou, X. (2023). A car model recognition method based on BP neural network. *Microelectronics and Computers*, (4).
9. Zhang, T. (2023). *Automobile model recognition based on neural networks* (Master's thesis). Beijing Institute of Technology.
10. Cui, J. (2023). *Research on image detection technology for driving vehicles* (Master's thesis). Nanjing University of Aeronautics and Astronautics.
11. Yao, L., Qin, C., Chen, Q., et al. (2020). Automatic extraction and recognition of road markings based on vehicle laser point cloud. *ISPRS Annals*, https://doi.org/10.5194/isprs-annals-V-2-2020-313-2020
12. Liu, F., Lu, Z., & Lin, X. (2025). Vision-based environmental perception for autonomous driving. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, *239*(1), 39–69. https://doi.org/10.1177/09544070231203059
13. Li, S., & Yang, S. (2023). Automatic dining car system based on A* algorithm. In *Proceedings of the 2nd International Conference on Computing Innovation and Applied Physics (Part 2)* (pp. 345–350). Xi'an Jiaotong Liverpool University. https://doi.org/10.26914/c.cnkihy.2023.109779
14. Hyla, T., & Wawrzyniak, N. (2021). Identification of vessels on inland waters using low-quality video streams. *Hawaii International Conference on System Sciences*.