МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Кафедра авіаційних комп’ютерно-інтегрованих комплексів

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач кафедри

Синєглазов В.М.

“ ____ ” __________2021.

ДИПЛОМНАРОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСНИКА ОСВІТНЬО-КВАЛІФІКАЦІЙНОГО РІВНЯ

“БАКАЛАВР”

Тема: Система відеоспостереження за контуром цілі

Виконав: **Гайда М.В.**

Керівник:**к.т.н.** **Василенко М. П.**

Нормоконтролер:к.т.н. **Тупіцин М. Ф.**

Київ – 2021

EDUCATION AND SCIENCE MINISTRY OF UKRAINE

NATIONAL AVIATION UNIVERSITY

COMPUTER-INTEGRATED COMPLEXES DEPARTMENT

ADMIT TO DEFENSE

Head of department

V. M. Sineglazov

“____” ______________ 2021.

BACHELOR WORK

(EXPLANATORY NOTES)

Topic: Video surveillance system of target contour

Done by:	**Haida M.V.**
Supervised by:	**Vasylenko M. P.**
Normcontrolled by:	**Tupitsyn M. F.**

Kyiv 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет аеронавігації, електроніки та телекомунікацій

Кафедра авіаційних комп’ютерно-інтегрованих комплексів

Освітній ступінь бакалавр

Спеціальність: 151 " Автоматизація та комп'ютерно-інтегровані технології"

ЗАТВЕРДЖУЮ

Завідувач кафедри

Синєглазов В.М.

“____” __________2021 р.

ЗАВДАННЯ

на виконання дипломної роботи студента

Гайди Максима Володимировича

1. **Тема проекту (роботи):** “ Система відеоспостереження за контуром цілі”
2. **Термін виконання проекту (роботи):** з 10.05.2021 р. до 11.06.2021 р.
3. **Вихідні данні до проекту (роботи):** метод виділення контуру цілі на відео, метод розпізнавання символів, метод визначення координат цілі, алгоритм програми, середовище Matlab.
4. **Зміст пояснювальної записки (перелік питань, що підлягають розробці):** 1. Актуальність системи відеоспостереження за контуром цілі; 2. Аналіз існуючих підходів для виявлення об'єктів; 3. Теоретичні основи системи відеоспостереження контуру цілі; 4. Вирішення задачі виявлення номерної таблички розробленою програмою та експерименти.
5. **Перелік обов’язкового графічного матеріалу:** 1. Структурна схема експериментальної установки; 2. Блок-схема алгоритму роботи програми; 3. Графічне зображення знайдених характерних точок; 4.

Графічне зображення відфільтрованих характерних точок ; 5. Папка із зображеннями шаблонів цифр та літер; 6.Початкове зображення із виділеною рамкою та символами.

6. Календарний план-графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1.	Отримання завдання	10.05.2021 – 11.05.2021	
2.	Формування мети та основних завдань дослідження	12.05.2021 – 13.05.2021	
3.	Аналіз існуючих методів	14.05.2021 – 19.05.2021	
4.	Теоретичний розгляд рішення задачі	20.05.2021 – 25.05.2021	
5.	Розробка структури системи відеоспостереження за контуром цілі	25.05.2021 – 30.05.2021	
6.	Розробка програмного та апаратного забезпечення системи відеоспостереження за контуром цілі	30.05.2021 – 05.06.2021	
7.	Оформлення пояснювальної записки	05.06.2021 – 07.06.2021	
8.	Підготовка презентації та роздаткового матеріалу	08.06.2021 – 11.06.2021	

7. Дата видачі завдання: “10” травня 2021 р.

Керівник дипломної роботи ____________________ М.П. Василенко

(підпис керівника) (П.І.Б.)

Завдання прийняла до виконання	____________________	М.В. Гайда
	(підпис випускника)	(П.І.Б.)

NATIONAL AVIATION UNIVERSITY

Faculty of aeronavigation, electronics and telecommunications

Department of Aviation Computer Integrated Complexes

Educational level bachelor

Specialty: 151 "Automation and computer-integrated technologies"

APPROVED

Head of Department

Sineglazov V. M.

"____" __________2021

TASK

For the student's thesis

Haida Maksym Volodymyrovych

1. **Theme of the project:** " Video surveillance system of target contour "
2. **The term of the project (work):** from May 10, 2021 until June 11, 2021
3. **Output data to the project (work):** the method of selecting the target contour on the video, the method of character recognition, the method of determining the coordinates of the target, algorithm of the program, the Matlab environment.
4. **Contents of the explanatory note (list of questions to be developed):**

1. The relevance of video surveillance system of target contour 2. Analysis of existing approaches of objects detection; 3. Theoretical basis of the video surveillance system of target contour; 4. Solution of number plate detection problem by the developed program and experiments.

5. **List of compulsory graphic material:**

1. Block diagram of the experimental setup; 2. Block diagram of the algorithm of the program; 3. Graphic representation of the found characteristic points; 4. Graphic representation of filtered characteristic points; 5. Folder with images of number and letter templates; 6. Initial image with a selected frame and symbols.

6. **Planned schedule:**

№	**Task**	**Execution term**	**Execution mark**
1.	Task	10.05.2021 – 11.05.2021	
2.	Purpose formation and describing the main research tasks	12.05.2021 – 13.05.2021	
3.	Analysis of existing methods	14.05.2021 – 19.05.2021	
4.	Analysis of existing systems	20.05.2021 – 25.05.2021	
5.	Theoretical consideration of the problem solution	25.05.2021 – 30.05.2021	
6.	Development of software and hardware for video surveillance system of target contour	30.05.2021 – 05.06.2021	
7.	Making an explanatory note	05.06.2021 – 07.06.2021	
8.	Preparation of presentation and handouts	08.06.2021 – 11.06.2021	

7. **Date of task receiving:** “10” May 2021

Diploma thesis supervisor Mykola P. Vasylenko

(signature)

Issued task accepted Maksym V. Haida

(signature)

CONTENT

GLOSSARY

AVSS	Automated video surveillance system
CV	Computer Vision
LTI	Linear Time Invariant
FIR	Finite impulse response
MSE	Mean square error
PSNR	Peak signal to noise ratio
CP	Characteristic points
FAST	Features from accelerated segment test
CSS	Curvature scale space
CNN	Convolutional Neural Networks
GPU	Graphics processing unit
CPU	Central processing unit
RAM	Random access memory
SSD	Sum of squared differences
CCORR	Cross-correlation
PC	Personal computer

INTRODUCTION

At a time when marauding, theft and various crimes are becoming more frequent, there is a great need in video surveillance systems installing. Such systems have already gained momentum and can be found in every supermarket or buildings under control.

However, the quality of video cameras and optical sensors in general is growing every year, as is the quality of the video they shoot, and it is no longer enough to simply record the video stream on a storage medium. There is a need to pre-process the video to efficiently use the storage memory and ensure automatic operation of the system without the participation of the operator.

This paper proposes an intelligent video surveillance system of the license plate contour, which will solve the problems described above.

Using a combination of methods for detecting objects by template matching, contour detection and feature detection in the image, was developed a system that is characterized by good accuracy and speed, can be both installed statically and be mobile, does not require expensive equipment and computing resources for its work, automatically detects the position of the number plate, highlights it, and recognizes the characters displayed on it.

Further in the course of work it is possible to know about the relevance of solving such a problem, already existing methods of the algorithm of work realization, theoretical data and experimental results of system work.

CHAPTER 1. THE RELEVANCE OF VIDEO SURVEILLANCE SYSTEM OF TARGET CONTOUR

1.1. Description of video surveillance systems

The rapid development of the functionality of video processing tools and their relatively low cost has led to the active use of digital technologies in various fields of human activity. In particular, computer video surveillance systems have become widespread for surveillance in banks, offices, airports, supermarkets, to search for subjects in the flow of people by appearance. In recent years, there has also been an active installation and use of video cameras at public transport stops, in parks, squares, schools, adjacent areas, etc. Such systems are increasingly used in forensics, access control systems, security systems [1].

In automated video surveillance systems (AVSS), the number of cameras is constantly growing and, accordingly, resource consumption increases. However, a significant limitation of such systems is the need for a large number of operators to service them. This led to the transition to intelligent AVSS. However, such systems reduce the efficiency of AVSS and require significant computing resources [2].

Video surveillance systems are one of the main components and occupy an important place in the overall structure of integrated security systems for facilities and individuals. In a world where crime, fighting, terrorist attacks and security breaches are on the rise, video surveillance systems are the right solution to prevent, detect and warn them [1].

Nowadays, video surveillance systems are increasingly integrated into various aspects of everyday life. One of the options for classifying such systems is the scope of their use:

- video surveillance of roads and highways: measuring the speed of cars, detecting driving at a red traffic light, crossing the dividing strip and other violations of traffic rules [1];

- public and commercial security: monitoring of public places to detect and prevent crime. These include individual facilities: schools, banks, supermarkets, theaters, department stores, parking lots, stadiums and entire transportation systems: airports, railways, subways, seaports, etc.;

- environmental monitoring and research: observation of forest fires and pollution, habitats and migration of animals, mountain ranges, plant diseases, oceanographic research, preservation of historical and archaeological monuments, cultural heritage;

- military sphere: patrolling state borders, measuring refugee flows, monitoring civilians, ensuring the security of military bases, assistance and management during hostilities, etc.;

- quality control: monitoring of industrial and automated processes, production sites to identify faults and intrusions into their infrastructure;

- smart homes and personal safety: home surveillance to prevent theft and intrusion, health of patients, children, animals, etc.;

- analysis of video information: determination of patterns and anomalies in the movement of transport, pedestrians, sports indicators, traffic in shopping malls, amusement parks, etc...

1.2. Computer vision and its tasks

Computer Vision (CV) is a field of artificial intelligence associated with image and video analysis. It includes a set of techniques that empower the computer to "see" and extract information from what it sees [3].

The systems consist of a photo or video camera and specialized software that identifies and classifies objects. They are able to analyze images (photos, pictures, videos, barcodes) as well as faces and emotions.

Machine learning technologies are used to teach a computer to "see". A lot of data is collected that allow you to identify features and combinations of features for further identification of similar objects.

All tasks of computer vision are reduced to the analysis of an image or video stream (in fact, it is a set of alternating images), on which it is required first of all to select a fragment containing the necessary information. For detection, is usually used a rectangular area, which limits the original fragment, or simply select the pixels belonging to it [4].

1.2.1. Identification

The identification task is to classify the whole image. To do this, key areas are highlighted in the image and classification is performed on them, for example, using decision trees, or convolutional neural networks.

1.2.2. Object recognition

The task is to be able to select a certain set of objects on the image. Until the problem is solved in the general case, the algorithm cannot classify random objects in the image. However, it is capable of recognizing a previously learned set of objects with a sufficiently high accuracy.

1.2.3. Image segmentation

The task is similar to object detection, but in contrast to it, it is required not to surround the found objects with frames, but to select the pixels that make up this object. Segmentation is used in many areas, for example, in manufacturing to indicate defects in the assembly of parts, in medicine for primary processing of images, as well as for compiling terrain maps from satellite images. One of the typical segmentation methods is the use of the U-Net model, which is several layers of a convolutional network that differ in size and are U-shaped in the stack, which is reflected in the name.

1.2.4. Pose Estimation

The problem of estimating the position of an object, in some way continues the task of segmentation. It consists in selecting a certain frame of the object (for example, a skeleton, if we are talking about people) and determining the position of this frame in the image. This skeleton can be used subsequently, for example, to predict the direction of movement. Depending on the number of objects under consideration, a Single-person pose estimation and Multi-person pose estimation are distinguished. The difference is that in the second case, it is also necessary to take into account that objects can overlap each other. To accomplish this task, the background is first cropped, leaving only the images of the objects themselves, and then for each of the objects, using convolutional neural networks, areas of joints are selected, which are then connected.

1.2.5. Text recognizing

One of the key tasks of computer vision. First, using detection algorithms, the area in which the text is written is highlighted, then the text is recognized directly, for example, using segmentation algorithms. At the same time, the tasks of recognizing text written on a sheet of paper and recognizing text written somewhere on the image (“in the wild”), for example, a text on a road sign, a car number, etc., are very different, due to the presence in the latter case interference that prevents you from detecting specific letters. In this case, for example, learning to predict a letter from the rest of the letters in a word can help.

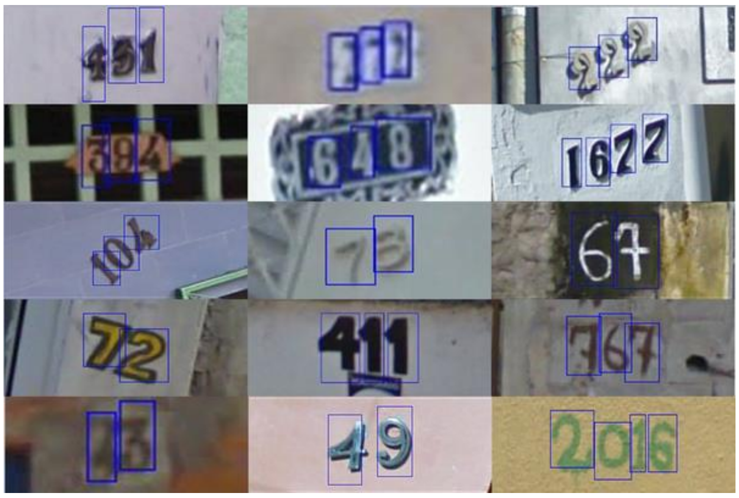


Fig.1.1. An example of a real-life text recognition task - Recognizing numbers on doors

1.2.6. Objects generation

The task is to learn how to create similar objects using a known set of objects, but at the same time they have not coincide with any of the test ones. For example, create animated characters in the style of a cartoon, drawing only a couple of them by hand. For this, such architectures are used as generative adversarial networks, in which the network is divided into two, one of which seeks to create an object, and the second to reject it, or a variational autoencoder that learns on the probability densities of the initial data in order to create an object similar to the original, but not the same.

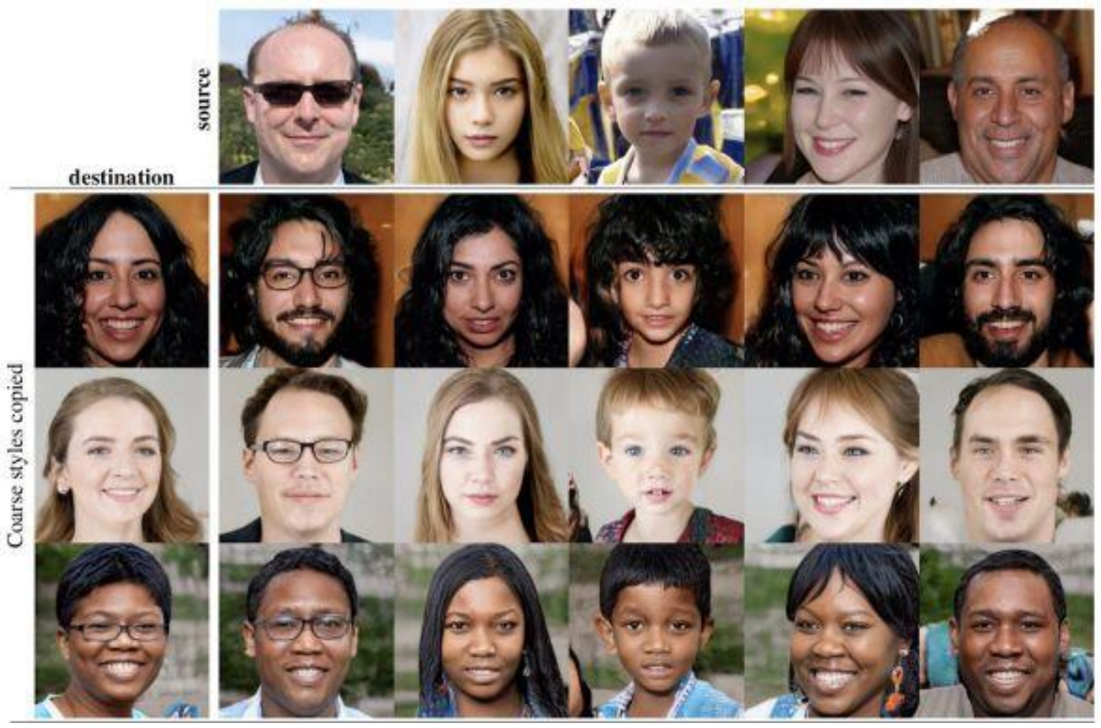


Fig. 1.2. An example of image generation using the GAN method

1.2.7. Video analysis

Since a video is a set of images of the same size, usually taken at different intervals of time, all the tasks that were described earlier are applicable to it. There are also tasks such as motion prediction, which consists in predicting the position of an object in the next frames from a set of frames, or a more general Situation Awarness task, which is to be able to determine its position for each object in a video. and status on all frames of the video.

1.3. The importance of detecting the target contour

In general, a video surveillance system is an information system consisting of video cameras, a complex of display and storage of video information, which is used to record, view and visually analyze video information. Video surveillance can be performed both in real time and viewing saved information from other storage [1].

In turn, the intelligent video surveillance system is a system with its own real-time operating system, which provides high reliability and makes the most efficient use of computer resources. Allows to achieve the maximum speed, the minimum reaction time to events and has long-term stability.

When recognizing objects, the most informative part of the image is the contour. An object contour is a part of an object that contains a lot of information about the shape of the object and depends little on the color and texture of the image.

The contour can be used to analyze the shape of the object. In many cases, information about the shape of the object is sufficient for the organization of automatic or automated systems. In addition, the transition to object recognition by their contours allows to reduce the amount of information processed by several orders, in addition, the contours are invariant to the brightness transformations.

Since the basic information about the shape of the object is contained in the contour of the object, the selection and description of the contour is an important task of image analysis.

1.4. Problem solving necessity

Nowadays, when video surveillance systems are installed at almost all controlled entrances to private territories (for example, parking lots, territories of government organizations, etc.), there is a great need to introduce systems with algorithms for tracking the contour of the target. That is, to track and recognize the license plate of a car that crosses the border of the controlled area, record information about the time of its entry and exit, the coordinates of the car and recognize the symbols located on the license plate and record them.

The first thing that comes to mind solving this problem is the application of the approach of neural networks deep learning. However, learning a neural network to recognize an object requires a large number of images from a training sample (thousands of examples), and the learning process also takes hours. And the most significant disadvantage of this approach is the process of its calculation on the chip of the video adapter, which already imposes restrictions on the use of hardware. Given the previous facts, such a system will be correspondingly expensive.

There is a second approach to solving the problem – Automatic Number Plate Recognition systems. They work fast and do not require large computing resources. However, in order for such a system to work properly, the camera is installed so that the car occupies the entire area of the frame, and the license plate is the largest rectangle in the frame. Such systems are very sensitive to changes in the position of the subject and various noises in the image, and recognition becomes unreliable if the background scenes of the frame occupy approximately more than 20% of the total image.

Intelligent video surveillance systems with the ability to track the target are now very expensive, with limited access to code and patented. The system shown in this paper has an open source code and a price that is determined only by the price of the camera and microcontroller.

Thus, there is a task to create a fast, inexpensive, easy to install system for tracking the contour of the license plate and recognition of license plates. What will be presented in this paper.

CHAPTER 2. ANALYSIS OF EXISTING APPROACHES OF OBJECTS DETECTION

There are many varieties of known methods for solving the problem of detecting the contour of the target. Some are based on the location and detection of moving objects in the video stream, while others detect the object by their belonging to a certain class and their characteristics. The following chapters will gradually describe the known methods of detecting objects in video and images.

2.1. Correlation method of objects detecting in the image

The task of detection is to establish the presence of objects (brightness areas) with certain properties in the image, and also, if the objects are detected, to determine their coordinates on the image plane [5].

The basic principle of object detection in an image is to compare the image brightness function with a certain "standard" - a fragment of the brightness field containing the desired object. When implementing the detection procedure, the standard fragment moves sequentially over the image field, and in each position its similarity with the real function of brightness on the fragment is examined. Complete coincidence of the standard and the image, as a rule, does not happen due to noise and distortions, and also due to the fact that there is usually no complete information regarding the shape and structure of the object (you have to use a standard that only approximately describes the object).

Let's denote by $t(k,l), (k,l)\in D$ the function of the brightness of the standard object, specified on a certain area D in the standard's own coordinate system (usually is considered a rectangular area D with the origin in the center). Let the $x(m,n)$ – be samples of the brightness function of the observed image. As a measure of the difference between the standard and the image at the point (m, n), the

ACIC DEPARTMENT				NAU 21 0217 000 EN			
Performed	Haida M.V.			VIDEO SURVEILLANCE SYSTEM OF TARGET CONTOUR	N.	Page	Pages
Supervisor	Vasylenko M.P.						
Consultant					431 151		
S. controller	Tupitsyn M.F.						
Dep. head	Sineglazov V.M.						

quadratic measure is most often taken:

$$\varepsilon^2(m,n) = \sum_{(k,l)\in D} [x(m+k,n+l) - t(k,l)]^2 \qquad (2.1)$$

It is considered that there is a similarity of the image fragment with the standard at the point *(m, n)*, if:

$$\varepsilon^2(m,n) < L_\varepsilon \qquad (2.2)$$

Where L_ε- some threshold depending on the intensity of the noise.

Fig.2.1. shows a schematic illustration of the principle of comparison with a standard.

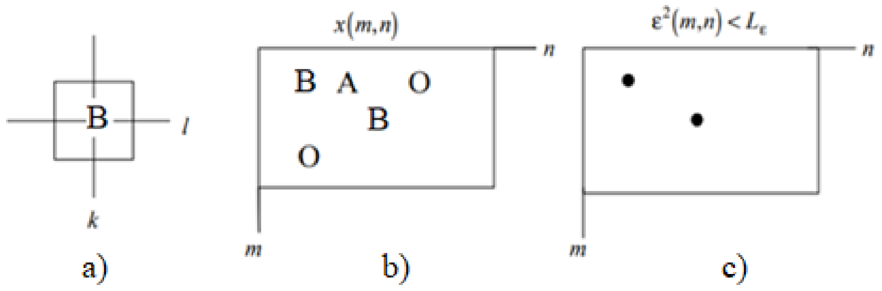


Fig.2.1. Illustration of the principle of comparison with a standard: a) a standard for comparison with; b) examined area; c) matches with the standard

In practice, the measure (2.1) is usually not calculated, but go over to related, but more simply calculated values. Transform the expression for the quadratic measure of difference:

$$\varepsilon^2(m,n) = \sum_{(k,l)\in D} x^2(m+k,n+l) - 2\sum_{(k,l)\in D} x(m+k,n+l)t(k,l) + \sum_{(k,l)\in D} t^2(k,l) \quad (2.3)$$

Here, the first term characterizes the image energy within the "window" D. This energy usually changes rather slowly depending on *(m, n)* and practically does not characterize the sought object. The third term characterizes the energy of the standard and does not depend on *(m, n)*. For detection, only the second term is essential, which, up to a constant factor, specifies the cross-correlation of the image and the standard [5]:

$$B(m,n) = \sum_{(k,l)\in D} x(m+k,n+l)t(k,l) \quad (2.4)$$

When the image and the reference coincide, the cross-correlation (2.4) is large, which ensures the smallness of the quadratic measure (2.1). However, direct use of cross-correlation as a measure of similarity usually results in poor detection performance. This is due to the fact that cross-correlation can increase even if the image does not match the standard if the brightness of the image in the vicinity of the point with coordinates *(m, n)* is high. This difficulty can be circumvented by using the normalized cross-correlation:

$$R(m,n) = \frac{\sum_{(k,l)\in D} x(m+k,n+l)t(k,l)}{\sqrt{\left[\sum_{(k,l)\in D} x^2(m+k,n+l)\right]\left[\sum_{(k,l)\in D} t^2(k,l)\right]}} \quad (2.5)$$

The value *R(m, n)* is equal to the maximum value (one) if the standard coincides with the image up to a constant non-negative factor. In this case, the object at the point *(m, n)* is considered detected if *R(m, n)>LR′*, where *LR′* is the threshold for measure (2.5).

The form of function (2.5) can be simplified if we first normalize the original image x (m, n) by choosing the energy of the standard equal to one, that is, to ensure $\sum_{(k,l)\in D} t^2(k,l) = 1$, performing adaptive element-by-element image processing with a "window" D to equalize the energy (variance) of the image

$\sum_{(k,l)\in D} x^2(m+k,n+l)$ all over the field. This further eliminates the influence of variance variations within the image. Denoting the normalized image by *x'(m,n)*, come again to the linear measure of cross-correlation:

$$B'(m,n) = R'(m,n) = \sum_{(k,l)\in D} x'(m+k,n+l)t(k,l) \qquad (2.6)$$

It is easy to see that the normalized cross-correlation function (2.6) is formed as a result of the passage of a two-dimensional signal - a normalized image *x'(m, n)* - through a two-dimensional LTI system with an impulse response *h(k,l)= t(-k,-l):*

$$B'(m,n) = x'(m,n) ** h(k,l) = x'(m,n) ** t(-k,-l) \qquad (2.7)$$

Thus, we obtain the general scheme of the correlation detection algorithm, which is shown in Fig. 2. Here, at the last stage, a binary image is obtained with ones at the points of object detection:

$$g(m,n)=\begin{cases}0, B'(m,n)\le L'_R,\\ 1, B'(m,n)>L'_R.\end{cases}$$

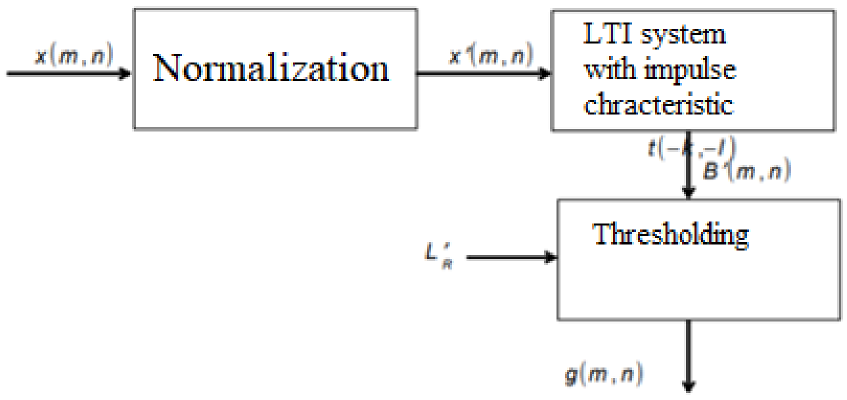


Fig. 2.2. Scheme of the correlation detection algorithm

LTI system, which in this case is called the correlator, is an FIR system and can be implemented by window (direct convolution) or spectral processing (fast convolution).

The considered correlation detection method is relatively simple, but it is characterized by rather high probability of errors (false detection or missing objects), which is explained by ignoring the properties of noise when synthesizing the image processing algorithm.

2.2. The method of object recognizing in the image by its contour

When recognizing objects, the most informative part of the image is the contour. An object outline is a part of an object that contains a lot of information about the shape of the object and depends little on the color and texture of the image.

The shape of the object can be analyzed along the contour. In many cases, information about the shape of the object is sufficient for the organization of automatic or automated systems. In addition, the transition to object recognition by their contours allows to reduce the amount of information processed by several orders of magnitude, in addition, the contours are invariant to the brightness transformations [1].

After digitization, each pixel uniquely refers to either the background or the image. There are different types of criteria for deciding whether each of the pixels belongs to the background or contour of the image.

The result of the selection of contours is an image skeleton - a secondary image of the same size as the original. Initially, all points of this image are black, and in the process of selecting the contours of the pixels that correspond to the detected boundary points of the image, become painted white.

The contour in the color image corresponds to the intensity difference. However, this definition excludes contours associated with abrupt changes in hue and intensity in areas with constant brightness.

Contour representation (encoding) is the step of obtaining a discrete signal that describes the boundaries of a digitized image.

Requirements for contour representation algorithms:

1. reducing the amount of memory used for storage;

2. reducing the time and complexity of further processing;

3. obtaining informative features of the object.

Biological systems of visual perception, as studies show, use mainly contours to highlight objects, rather than dividing objects by brightness. In practice, the differences will not be sharp due to the blurring and limitations imposed by the video recording equipment. Sometimes the brightness differences along the boundaries are better traced in the form of jumps in the first brightness derivatives than in the analysis of the values of the brightness itself.

When solving the problem of contour selection, a compromise between the number of erroneous contours and the number and magnitude of contour breaks is found. It is known that the result of the follow-up operation is much less affected by small gaps. They are easier to eliminate than false contours, in which it is easy to get confused [1].

The relationship between the number of false contours and the number and magnitude of gaps is determined by the noise immunity of the method of contour selection. Any region D of the plane of the complex variable contains internal points and contour points (boundary points). The first of them have the property that not only they themselves, but also some of their surroundings belong entirely to the area D. Contour points are not internal, but in any small neighborhood of such points there are internal points of area D and points which do not belong to area D - external (background) points. Area D has the property of connectivity, which is that any of its points are connected by a line that is completely in the middle of D.

A contour line G is said to be convex if the rectilinear segment connecting any two of its points consists entirely of the interior points of region D. The contour section will be concave if such a segment will include external (background) points (Fig. 2.3.).

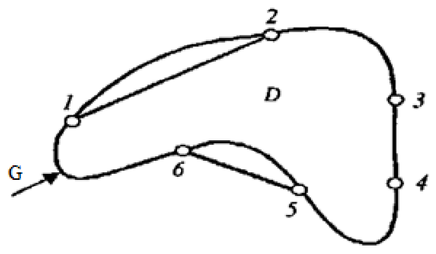


Fig. 2.3. Fragments of the contour G: 1, 2 - convex; 3, 4 - indefinite; 5, 6 - concave

Internal element (pixel) of a binary digitized image *ω*(m1,m2) has the property of four-connectivity, ie adjacent elements - upper, lower, left and right, also belong *ω* (m1,m2).

To process the contour in an analytical way, it is necessary to encode it, ie to set a certain number in accordance with each contour element. The sequence of such numbers is called the contour code. Eight different standard placements are possible on the square grid.

Let’s consider some ways to encode contours:

1. Coding by three features: the length of the current elementary vector, the direction of rotation when moving to the next elementary vector and the angle between adjacent elementary vectors.

2. Encoding the current elementary vector with a three- dimensional binary code (numbers from 0 to 7). This code was proposed by Freeman and is widely used in image processing.

3. The coding of the current elementary vector by its two projections on the coordinate axis with the origin, combined with the beginning of the elementary vector - two-dimensional code.

4. Polygonal representation of the contour obtained by its approximation by linear segments (Fig. 2.4).

Coding is to fix the coordinates of the ends of these segments. This method due to the compactness of the obtained descriptions has become widespread. This causes a segmentation problem similar to the signal sampling problem. In real cases, it is usually associated with the loss of information about the shape of the images.

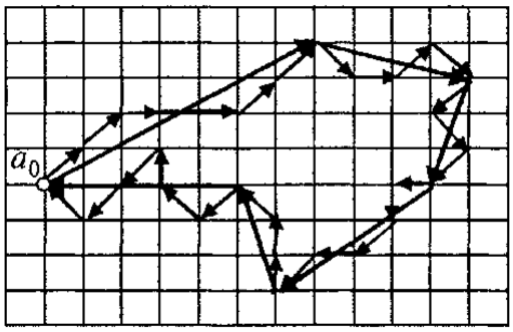


Fig. 2.4. Polygonal representation of a contour by means of approximation of a contour by linear segments

5. Representation of a contour line by a polar code. In the image ω(m1,m2) the pole is selected - the beginning of the reference (point P) ordinary (own) coordinate system, ie the frame of reference associated with this image. The centers of all boundary points of the image are connected with the point P. The result is a sequence of radius vectors β (n), that uniquely define the contour of the image (Fig. 2.5). Often the center is aligned with the center of gravity of the image.

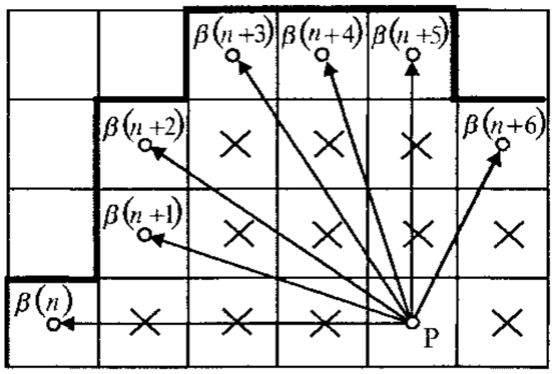


Fig. 2.5. An example of specifying a fragment of the contour in the polar coordinate system

Methods of contour selection can be divided into two large groups: differential and extreme correlation. In differential methods, the intensity differences are amplified by numerical differentiation, then the contour is selected by a threshold device, after which the binary image is subjected to secondary processing, the purpose of which is to thin the contour to one pixel. The methods are easy to implement and have high performance, but have low noise immunity. The main criterion in assessing the noise immunity of the contours is the position of the brightness difference.

On the other hand, two approaches are used to define and describe a contour: selecting edges or selecting the area of a point that forms an object.

A large number of algorithms for selecting contours and boundaries are given and described in the literature. The most popular methods include the Roberts, Sobel, Previtt, Kirsch, Robinson operator, the Canny algorithm and the LoG algorithm. These algorithms are based on emphasizing the sharp differences in brightness that are characteristic of the contours of objects.

2.2.1. Roberts operator

The Roberts operator is one of the first contour selection algorithms that calculates the sum of the squares of the differences between diagonally adjacent pixels. This will be an image convolution with two cores:

$$\begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix} and \begin{bmatrix} 0 & +1 \\ -1 & 0 \end{bmatrix} \tag{2.8}$$

Roberts' operator is still used for computing speed, but it loses compared to alternatives due to its significant noise sensitivity. It makes the lines thinner than other methods of contouring, which is almost equivalent to calculating the final differences along the X and Y coordinates. It is sometimes called the "Roberts filter".

2.2.2. Laplace operator

The discrete Laplace operator is also often used in image processing to highlight contours. Discrete Laplacian is defined as the sum of the second derivatives and is calculated as the sum of the differences on the "neighbors" of the central pixel. For a one-dimensional signal, a discrete Laplacian can be written as a convolution with the next core:

$$D_x^2 = [1 \ -2 \quad 1] \tag{2.9}$$

And for a two-dimensional signal:

$$D_x^2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \ or \ D_x^2 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & -8 & 1 \\ 1 & 1 & 0 \end{bmatrix} \tag{2.10}$$

2.2.3. Prewitt operator

The Prewitt operator is a method of selecting contours in image processing, which calculates the maximum deviation on the set of convolution cores to find the local orientation of the contour in each pixel. It was created by Dr. Judith Prewitt to identify the contours of medical images.

The operator uses two cores 3 × 3, convolving the original image to calculate the approximate values of the derivatives, one horizontally and the other vertically:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix} \ and \ G_x = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{bmatrix} \tag{2.11}$$

2.2.4. Sobel operator

The Sobel operator is a discrete differential operator that calculates the approximate value of the image gradient. It is used in the field of image processing, in particular, often used in contour selection algorithms. The result of applying the Sobel operator at each point of the image is either the brightness gradient vector at this point, or its norm, calculated by formulas:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * A \; and \; G_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * A \qquad (2.12)$$

where *A* is the input image.

2.2.5. Canny operator

Canny studied the mathematical problem of obtaining a filter, according to the optimal criteria for selection, localization and minimization. He showed that the desired filter is the sum of four exponents and can be well approximated by the first Gaussian derivative. Canny introduced the concept of Non-Maximum, which means that the pixels of the contours are declared pixels in which the local maximum of the gradient in the direction of the gradient vector is reached. Although his work was carried out in the early days of computer vision, Kanny's contour detector is still one of the best detectors.

The result of the described algorithms is a set of incoherent areas. To obtain a connected contour, additional processing is required, such as morphological processing to obtain the connected edge of the object, which is called the contour of the object.

2.2.6. Comparative analysis of the described methods of contour selection

The comparative analysis of the described methods of contour selection is carried out. In comparative analysis, the assessment of image quality is widely used. This estimate is characterized by a number of metrics that show how exactly the resulting image matches the original. The best known metrics are the mean square error (MSE) and peak signal to noise ratio (PSNR).

The mean square error (MSE) is an indicator of the dispersal of values of a random variable relative to its mathematical expectation:

$$MSE = \frac{\sum_{i=1,j=1}^{m,n}\left(x_{i,j} - y_{i,j}\right)^2}{mn} \tag{2.13}$$

where *MSE* – the mean square error; $x_{i,j}$ – sampling element; $y_{i,j}$ – arithmetic mean of the sample; *n*, *m* is the sample size.

The peak signal to noise ratio (PSNR) is the ratio between the maximum possible signal value and the noise power that distorts the signal value. The easiest way to determine this ratio is using mean square error:

$$PSNR = 10log_{10}\left(\frac{MAX_I^2}{MSE}\right) \tag{2.14}$$

where MAX_I is the maximum value of the pixel intensity in the image; *MSE* is the mean square error.

The higher the value of the peak signal-to-noise ratio, the clearer and more accurate the image is considered. To evaluate the quality of the obtained image with the selected contours by the described methods and to compare them, the image from the ORL database was used. This image will be considered the original, ie in our case it is a perfect image.

Also, an important indicator of the operation of contour selection methods is the speed, which is measured in seconds. Table 1 shows the results.

Table 2.1. Comparison of the performance speed of known methods of contour selection

Image contour selection method	Performance speed, sec
Roberts	0,60
Laplace	0,86
Prewitt	0,63
Sobel	0,76
Canny	2,45

As can be seen from the results, the Canny method gives the best results in terms of standard deviation and peak signal-to-noise ratio, but it is the slowest of the considered methods.

Thus, it is shown that the basic information about the object is contained in the contour, so the selection and description of the contour is an important task of image analysis, contours are invariant to brightness transformations, and the transition to object recognition by their contours allows to reduce the amount of processed information.

A comparative analysis of contour selection methods was also performed. Analysis of the speed, standard deviation and peak ratio of signal to noise showed that the Canny method has the lowest contour selection speed, but according to the criteria of standard deviation and peak signal to noise shows the best results.

2.3. The method of recognizing an object in an image by characteristic points

2.3.1. General information about the method

Characteristic points (CP) is an image point with high local informativeness, ie it is the points of maximum, minimum, inflection and maximum curvature. CP are also called salient, keypoints, representative, feature points, characteristic points, inflection points.

Examples of CP are: the ends of the segment, the vertices of the polygons, the inflection points, the inflection points of the splines, the end points of the semi-axes of the ellipse.

In 1992, Haralick and Shapir identified the following requirements for CP:

- distinctness - CP should stand out clearly in the background and be unique in its surroundings;

- invariance - the detection of CP should be independent to affine transformations;

- stability - the detection of CP should be resistant to noise and errors;

- uniqueness - in addition to local differences, CP must have global uniqueness in order to improve the distinction of repetitive elements;

- interpretability - CPs should be defined so that they can be used to analyze matches and identify interpreted information from an image.

Tuytelaars and Mikolajczyk (2006) described the properties that CPs should have:

- repeatability - CP is in the same place of the scene or image object, despite changes in the point of view and lighting;

- distinctiveness / informativeness - there should be an environment of CP where it should have big differences from other points in this neighborhood so that it was possible to allocate and compare special points;

- locality - CP should occupy a small area of the image to reduce the likelihood of sensitivity to geometric and photometric distortions between two images taken at different points of view;

- quantity - the number of detected CP should be large enough so that they are enough to detect even small objects. However, the optimal amount of CP depends on the subject area. Ideally, the number of detected CP should be adaptively determined using a simple and intuitive threshold.

- location density - CP should display the information content of the image to ensure its compact presentation;

- accuracy - detected CP must be accurately localized, both in the original image and in taken at another scale one;

- efficiency - the time of detection of CP in the image should be acceptable in time-critical applications.

In general, these properties intersect with the previous ones, but are interpreted differently.

The main advantage of using CP for identification tasks is the relative simplicity and speed of their detection.

The class of methods for finding key points is called "keypoint detection", and algorithms for comparing and searching for images using key points - "keypoint matching". Searching for a pattern in a picture comes down to applying the key point detection algorithm to the pattern and the picture, and matching the key points of the pattern and the picture [6].

Usually “key points” are found automatically by finding pixels whose surroundings have certain properties. Many methods and criteria for finding them have been invented. All these algorithms are heuristics that find some characteristic elements of the image, as a rule - corners or sharp drops in color.

The process of identifying special points is achieved through the use of a detector and a descriptor.

A detector is a method for extracting specific points from an image. The detector ensures the invariance of finding the same singular points with respect to image transformations.

Descriptor - an identifier of a special point, which distinguishes it from the rest of the set of special points. In turn, descriptors must ensure the invariance of finding a correspondence between singular points with respect to image transformations.

A good detector should work quickly and be resistant to image transformations (when the image changes key points detection should not stop / moving).

2.3.2. Corners detectors

Corners are special points that are formed from two or more facets, and facets usually define the border between different objects and/or parts of the same object. The main property of such points is that two dominant directions prevail in the area around the corner of the image gradient, which makes them distinguishable [7].

Gradient is a vector value that shows the direction of the steepest increase in the function of the image intensity *I(x,y)*. Since the image is discrete, the gradient vector is determined through partial derivatives along the *x* and *y* axes through changes in the intensities of neighboring points of the image. Most of the methods consider angularity depending on the 2nd order derivative, therefore, in general, the methods are sensitive to noise.

Depending on the number of intersecting facets, there are different types of corners: L-, Y- (or T-), and X-connected (some also distinguish arrow-connected angles). Different corner detectors react differently to each of these corner types.

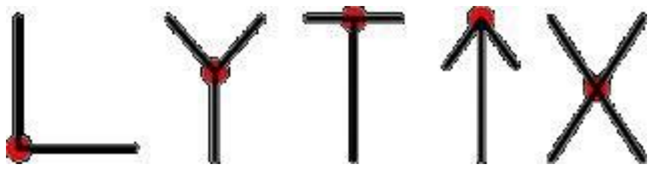


Fig. 2.6. Corner types

The approaches of identifying special points can be divided into 3 categories:

1. Based on image intensity: the feature points are calculated directly from the pixel intensities of the image.
2. Using image contours: methods extract contours and look for places with maximum curvature or make a polygonal approximation of contours and determine intersections. These methods are sensitive to neighborhood intersections, as extraction can often be incorrect where 3 or more edges intersect.
3. Model Based: uses models with intensity as parameters which adjust to template images with subpixel precision. They have limited use with special types of feature points (for example, L-connected corners), depend on the templates that are used.

In practice, for widespread use, the most common methods based on image intensity are used.

Next, I will briefly discuss the pros and cons of basic corner detectors. Then a comparative table of detectors will be presented with conclusions about their applicability to different situations.

2.3.3. Moravec corner detector general description

Moravec detector - the simplest of the existing. The author examines the change in the brightness of a square window W (usually 3x3, 5x5, 7x7 pixels) relative to the point of interest when the window W is shifted by 1 pixel in 8 directions (horizontal, vertical and diagonal).

The Moravec detector has the property of anisotropy in 8 directions of window displacement. The main disadvantages of the considered detector are the lack of invariance to the rotation transformation and the occurrence of detection errors in the presence of a large number of diagonal edges.

2.3.4. Harris corner detector general description

Harris and Stephens improved the Moravec detector by introducing anisotropy in all directions, i.e. consider the derivatives of the brightness of the image to study changes in brightness in many directions. They introduce derivatives in some fundamental directions.

The Harris detector is rotational invariant, partially invariant to affine intensity changes. The disadvantages include the sensitivity to noise and the dependence of the detector on the image scale (to eliminate this disadvantage, a multi-scale Harris detector is used).

2.3.5. Shi-Tomasi corner detector general description

The Shi-Tomasi angle detector (Shi-Tomasi or Kanade-Tomasi, 1993) is largely the same as the Harris detector, but differs in the computation of the response measure: the algorithm computes the value directly because it makes the assumption that the search for corners will be more stable. The authors use the same equation to analyze the optical flow of Lucas and Kanade.

2.3.6. Förstner corner detector general description

Förstner and Gülch (1987) were the first who described a method that uses the same measure of angularity as the Harris detector. They used a more computationally complex implementation. Unlike the Harris detector, the eigenvalues are calculated explicitly. The Förstner angle response function is defined as follows:

$$R = \lambda_1\lambda_2/(\lambda_1 + \lambda_2) = \frac{detM}{trM} \tag{2.15}$$

Also, for the correctness of the definition, the measure of the roundness of the angle is considered, equal to:

$$1 - \left(\frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2}\right)^2 = \frac{4detM}{(trM)^2} \tag{2.16}$$

The Förstner detector is often used in practice to expand the capabilities of the Harris detector - finding circular feature points along with angles. Also, the algorithm has the best localization property.

2.3.7. SUSAN corner detector general description

Angles are defined by segmentation of circular neighborhoods into similar (orange) and dissimilar (blue) areas (Fig. 2.7). The corners are located where the relative area of similar areas (similar USAN) reaches a local minimum below a certain threshold.

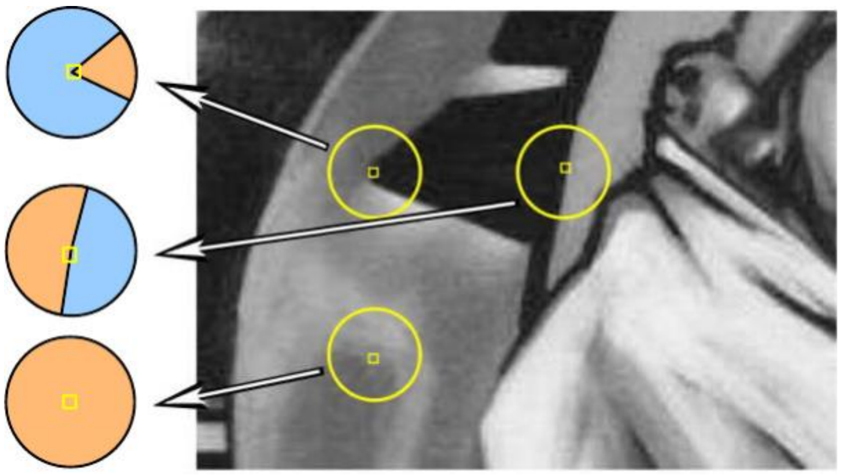


Fig. 2.7. SUSAN detection algorithm visualisation

The algorithm shows good accuracy for all kinds of angles, but is not resistant to blur in images.

2.3.8. Trajkovic corner detector general description

The detector checks the area around a pixel by examining nearby pixels: let *c* be the pixel to be examined and *P* the point on the circle S_N at the center at point *N*. Point *P'* is a point opposite *P* in diameter.

In comparison with the Harris detector, the repetition rate of the Trajkovic4 algorithm is worse, but the localization is comparable to the determination of L-connected angles and is superior in other types of angles.

Also disadvantages include the fact that this 4-adjacent operator reacts falsely to diagonal edges and is sensitive to noise. Therefore, an 8-connected version of this Trajkovic8 algorithm is used. Trajkovic8 differs from Trajkovic4 in how it calculates angularity. However, Trajkovic8 still finds false angles on some of the diagonal edges of the object (it doesn't work well on artificial images).

2.3.9. FAST corner detector general description

An alternative to Harris's method is FAST. As the name suggests, FAST is much faster than the mentioned method. This algorithm tries to find points that lie on the edges and corners of objects, i.e. in places where there is a difference in contrast. They are found as follows: FAST builds a circle of radius *R* around the candidate pixel and checks if there is a continuous segment of pixels of length *t* on it, which is *K* units darker (or lighter) than the candidate pixel. If this condition is met, then the pixel is considered a “key point” [6].

The main disadvantage of the algorithm is that several singular points can be found near a certain neighborhood; the efficiency of the algorithm depends on the order of image processing and the distribution of pixels. But it works fast enough in comparison to competitors.

2.3.10. CSS corner detector general description

Rattarangsi and Chin (1992) proposed a curvature scale space (CSS) algorithm that detects corners on planar curves. CSS is suitable for extracting invariant geometric features on a flat curve at various scales.

Algorithm identifies feature points using multiple scales of the same image. However, it is computationally complex and detects false angles in circular areas.

This algorithm has the following disadvantages: an image with only one scale is used to determine the number of angles, and images at multiple scales are used for localization. As a result, the algorithm skips corners when σ is large and detects false angles when σ is small.

2.3.11. Corner detectors comparison

Table 2.2. Comparison of angle detectors (where 1 - Very bad, 2 - poor, 3 - fair, 4 - good, 5 - excellent)

Operator (algorithm)	Detection efficiency	Localization	Repetition frequency	Noise resistance	Speed
Moravec	3	4	3	3	4
Förstner	4	4	5 for affine transformations, 3 for scaling	4	2
FAST	4	4	5	4	5
Harris	4	4 for L-connected angles, 2 for other types	5 for affine transformations if anisotropic gradient is computed, 3 for scaling	3	2
SUSAN	4	1 for blurry images, 4+ otherwise	4 for scaling, 2 for affine transformations	5	4
CSS	4	4	5	4	Strongly depends on the contour detector
Trajkovic & Hedley	2	4	3 (not invariant to rotations)	2	5

After analyzing the available methods, it was decided to use the FAST angle detector in algorithm because of its ease of use, the method for finding CPs and, most importantly, speed.

2.4. Object recognition using neural networks (deep learning) and machine learning

It is possible to use various approaches for object recognition. Recently, machine learning and deep learning techniques have become popular approaches to object recognition problems. Both technologies learn to recognize objects in images, but they differ in their performance [8].

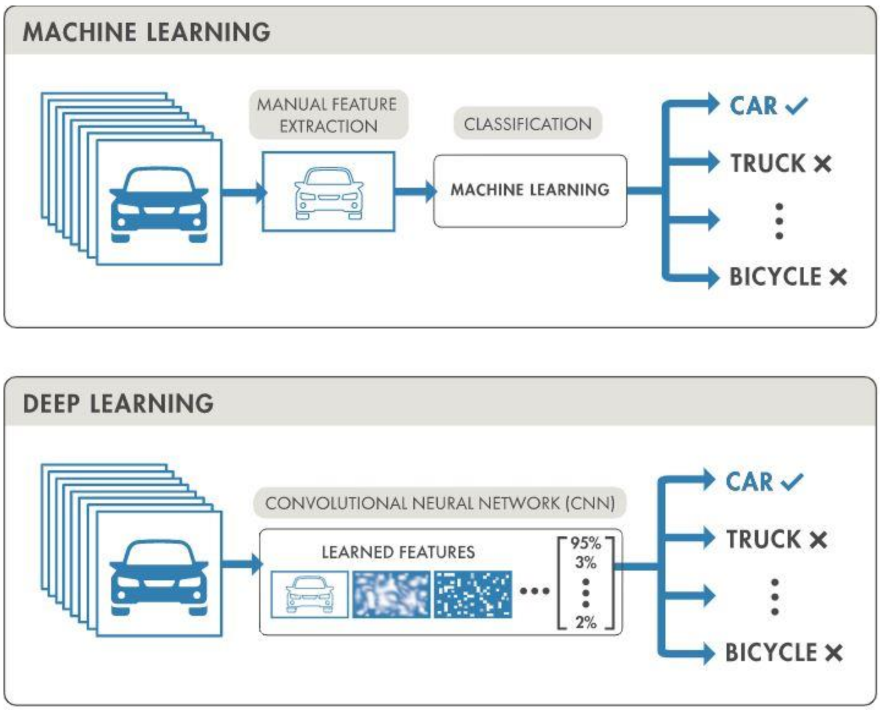


Fig. 2.8. Machine learning and deep learning techniques for object recognition

2.4.1. Object recognition using deep learning general description

Deep learning techniques have become a popular method of object recognition. Deep learning models such as Convolutional Neural Networks (or CNNs) are used to automatically learn the inherent properties of an object in order to identify that object. For example, CNN can learn to distinguish between cats and dogs by analyzing thousands of images and studying the characteristics that distinguish cats and dogs.

There are two approaches to object recognition using deep learning:

- Model training from scratch. To train a deep network from scratch, you need to collect a very large labeled dataset and develop a network architecture that learns characteristics and builds a model. The results can be impressive, but this approach requires a lot of training data and also requires setting the levels and weights in CNN.

- Using a pretrained deep learning model: most deep learning applications use a transfer learning approach - a process that involves fine-tuning of pre-trained model. To begin with, an existing network such as AlexNet or GoogLeNet is taken and new data containing previously unknown classes is entered. This method is less time consuming and can provide faster results since the model has already been trained on thousands or millions of images.

Deep learning offers a high level of accuracy, but requires a lot of data to make accurate predictions.

2.4.2. Object recognition using machine learning general description

Machine learning techniques are also popular for object recognition and offer different approaches than deep learning. Common examples of machine learning techniques are:

- extracting HOG features using SVM machine learning model;

- bag-of-words models with functions such as SURF and MSER;

- Viola-Jones algorithm, which can be used to recognize various objects, including faces and upper part of the body.

To recognize objects using the standard machine learning approach, you need to start with a set of images (or video) and select the appropriate characteristic in each image. For example, the feature extraction algorithm can extract edge or corner features that can be used to distinguish classes in user data. These features are added to a machine learning model that separates these characteristics into separate categories, and then uses this information to analyze and classify new objects. Variety of machine learning algorithms and feature extraction techniques can be used that offer many combinations to create an accurate object recognition model.

Using machine learning for object recognition gives you the flexibility to choose the best combination of features and classifiers for training. It can achieve accurate results with minimal data.

2.4.3. Machine learning and deep learning for object recognition comparison

The choice of the best approach to object recognition depends on the task at hand. In most cases, machine learning can be an effective method, especially if you know which image characteristics are best used to distinguish classes of objects.

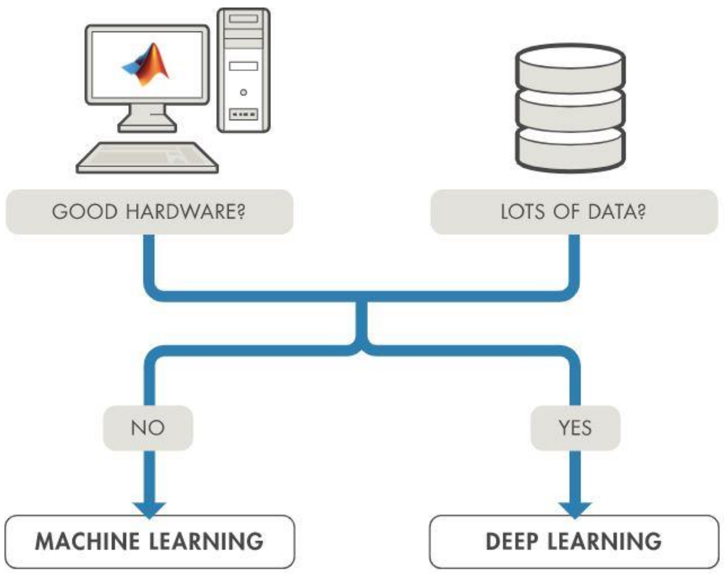


Fig. 2.9. Key factors in choosing between deep learning and machine learning

The main thing to keep in mind when choosing between machine learning and deep learning is having a powerful GPU and lots of labeled training images. If the answer to any of these questions is no, a machine learning approach may be the best choice. Deep learning techniques tend to work better with large numbers of images, and the GPU helps to reduce the time it takes to train the model.

CAPTER 3. THEORETICAL BASIS OF THE VIDEO SURVEILLANCE SYSTEM OF TARGET CONTOUR

A car was chosen as the object of surveillance for the program. All cars have common features such as headlights, windshield, rear-view mirrors, wheels, license plate. However, for each of the existing cars, the models of which there are a very large number, almost all of these characteristics differ in size, shape and even location.

The task of object recognition, with taking into account all the above-described features, could be solved only by neural networks. However, for this they need thousands, or even hundreds of thousands of images of the training sample and large computing power. Therefore, it was decided to develop own program that identifies the car on the video stream, namely its license plate. The license plate was chosen as the object of tracking, because it is the feature that has constant proportions, is present on all cars, and is the best object to track it.

3.1. Blurring by Gaussian filter

Blurring is an integral part of various image correction techniques aimed at eliminating specific defects (excessive detail, scan defects, dust, etc.). One of their possible applications is noise reduction, i.e. the problem of restoring the original image with random noise added to its pixels [9].

Gaussian blur is a generic image blur filter that uses a normal distribution (Gaussian distribution) to compute the transform applied to each pixel in an image. The noise in the image changes independently from pixel to pixel and if the mathematical expectation of the noise value is equal to zero, the noise of neighboring pixels will compensate each other. The larger the filtering window, the less the average intensity of the noise will be, however, significant blurring of significant image details will also occur.

The Gaussian distribution equation in *N* dimensions has the form:

$$G(r) = \frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} e^{-\frac{r^2}{2\sigma^2}} \qquad (3.1)$$

Noise reduction using a rectangular filter has a significant drawback: pixels at a distance "r" from the processed one have the same effect on the result as neighboring ones.

Thus, more effective noise reduction can be realized if the influence of pixels on each other decreases with distance (a special case - for two dimensions):

$$G(u.v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}} \qquad (3.2)$$

where *r* is the blur radius, *r2 = u2 + v2*, *y* is the standard deviation of the Gaussian distribution.

In the case of two dimensions, this formula defines a surface that looks like concentric circles with a Gaussian distribution from the center point. Pixels where the distribution is nonzero are used to construct a convolution matrix that is applied to the original image. The value of each pixel becomes a weighted average for the neighborhood. The original pixel value gets the biggest weight (has the highest Gaussian value), and neighboring pixels get less weights, depending on the distance to them. In theory, the distribution at each point in the image will be nonzero, which would require the calculation of weighting factors for each pixel in the image. But, in practice, when the discrete approximation of the Gaussian function is calculated, pixels at a distance of more than 3 are not taken into account, since they are small enough. Thus, it is enough for the program filtering the image to calculate the matrix in order to guarantee sufficient accuracy in the approximation of the Gaussian distribution.

To apply this filter, convolution by function is used:

$$I'(i,j) = \sum_{I=-n}^{n} \sum_{k=-m}^{m} I(i-l)(j-k) * \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-d^2}{2\sigma^2}} \quad (3.3)$$

The parameter *y* sets the degree of blur. On the graph, a function with *y = 5*

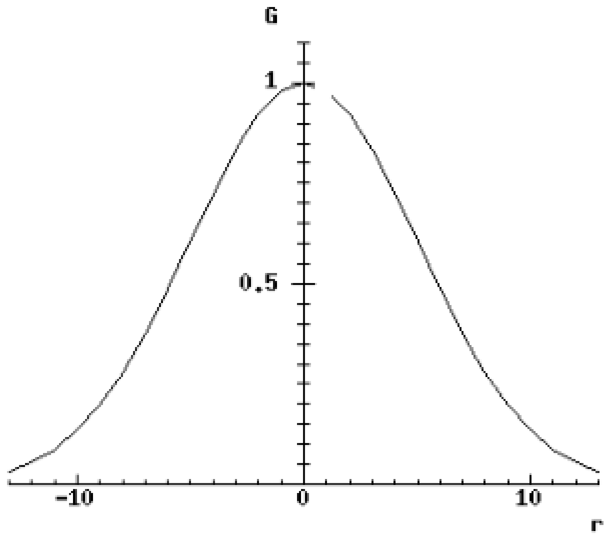


Fig. 3.1. Graph of Gaussian function

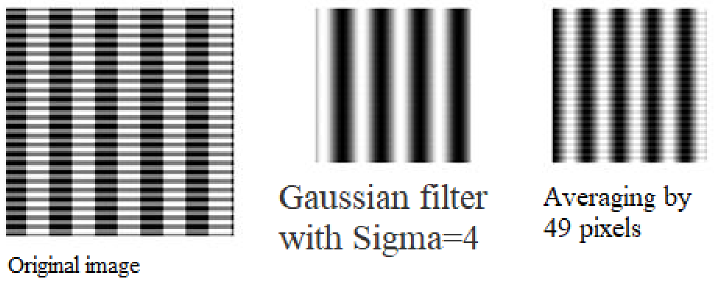


Fig. 3.2. The results of convolution by the Gaussian function and by a constant function (averaging).

The Gaussian filter is well suited for a situation where a noisy image has a large amount of details, because this filter blurs small details less and removes noise quite adequately.

3.2. FAST characteristic points recognizing algorithm description

FAST, was first proposed in 2005 in the work, was one of the first heuristic methods for finding special points, which gained great popularity due to its computational efficiency [10]. To decide whether to consider a given point *C* special or not, this method considers the brightness of pixels on a circle with a center at point *C* and a radius of 3:

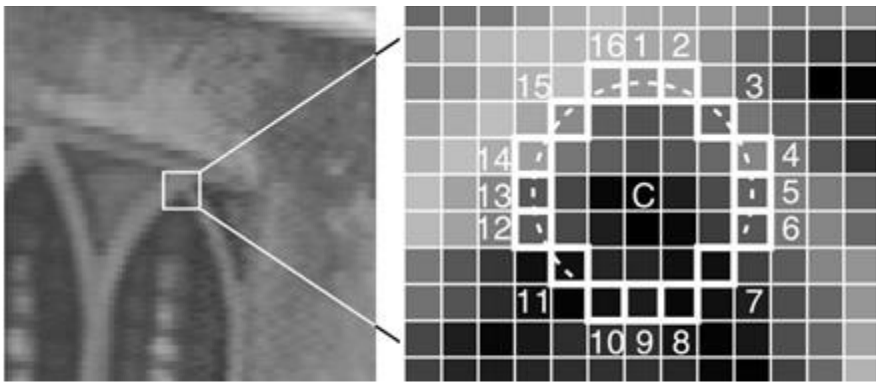


Fig. 3.3. Pixels considered by FAST detector

Comparing the brightness of the pixels of the circle with the brightness of the center *C*, we get for each three possible outcomes (lighter, darker, it seems):

$$\begin{gathered} I_p > I_C + t \\ I_p < I_C + t \\ I_C - t < I_C < I_C + t \end{gathered} \qquad (3.4)$$

here *I* is the brightness of pixels, *t* is some predetermined brightness threshold.

A point is marked as special if there are n = 12 pixels in a row on the circle that are darker, or 12 pixels that are lighter than the center.

As practice has shown, on average, to make a decision, it was necessary to check about 9 points. In order to speed up the process, the authors proposed to first check only four pixels numbered: 1, 5, 9, 13. If among them there are 3 pixels lighter or darker, then a full check is performed by 16 points, otherwise - the point is immediately marked as "not special". This greatly reduces the operating time; to make a decision, on average, it is enough to interrogate only about 4 points of the circle.

Initially, the original algorithm was FAST-12. There are modifications of the algorithm: the tree based FAST-9 and FAST-12.

The original algorithm has a number of disadvantages, for example, several special points may be found near a certain neighborhood, the efficiency of the algorithm depends on the order of image processing and the distribution of pixels.

Edward Rosten, Reid Porter, and Tom Drummond (2008) introduce improvements to the FAST algorithm in that they use machine learning to identify feature points.

They called this algorithm FAST-ER (ER - Enhanced Repeatability). The algorithm is stable to the property of repeatability: on the same scene, viewed from different angles, there are feature points belonging to the same objects.

This algorithm uses a circle of more than 1 pixel unlike FAST (48 pixels). The authors use the ID3 algorithm to classify feature points (whether a candidate point is characteristic) using decision trees. ID3 algorithm optimizes the order in which pixels are processed, resulting in the most computationally efficient detector.

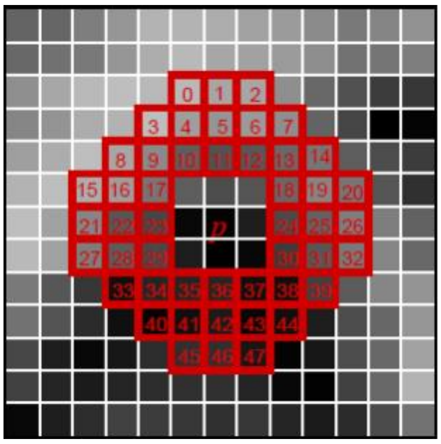


Fig. 3.4. Pixels considered by FAST-ER detector

The decision tree cost function is calculated as follows:

$$cost = (k_R + R^{-2})(k_N + N^{-2})(k_S + S^{-2}) \qquad (3.5)$$

Where *R* is a measure of repeatability; *N* is the number of detected feature points; *S* is the number of nodes in the decision tree.

FAST-ER is better than FAST, but slower in execution speed. The authors concluded that the FAST-ER detector is the best in terms of repeatability.

Through experiments on images, it was determined that the FAST algorithm fully meets the needs of the program, and was chosen for use.

3.3. Selection of contours by the Prewitt operator description

All known methods are based on one of the basic properties of the brightness signal - discontinuity. The most common way to find gaps is to process an image using a sliding mask, also called a filter, kernel, window, or pattern, which is a kind of square matrix corresponding to a specified group of pixels in the original image. Matrix elements are usually called coefficients. Operating with such a matrix in any local transformations is called filtering or spatial filtering [11].

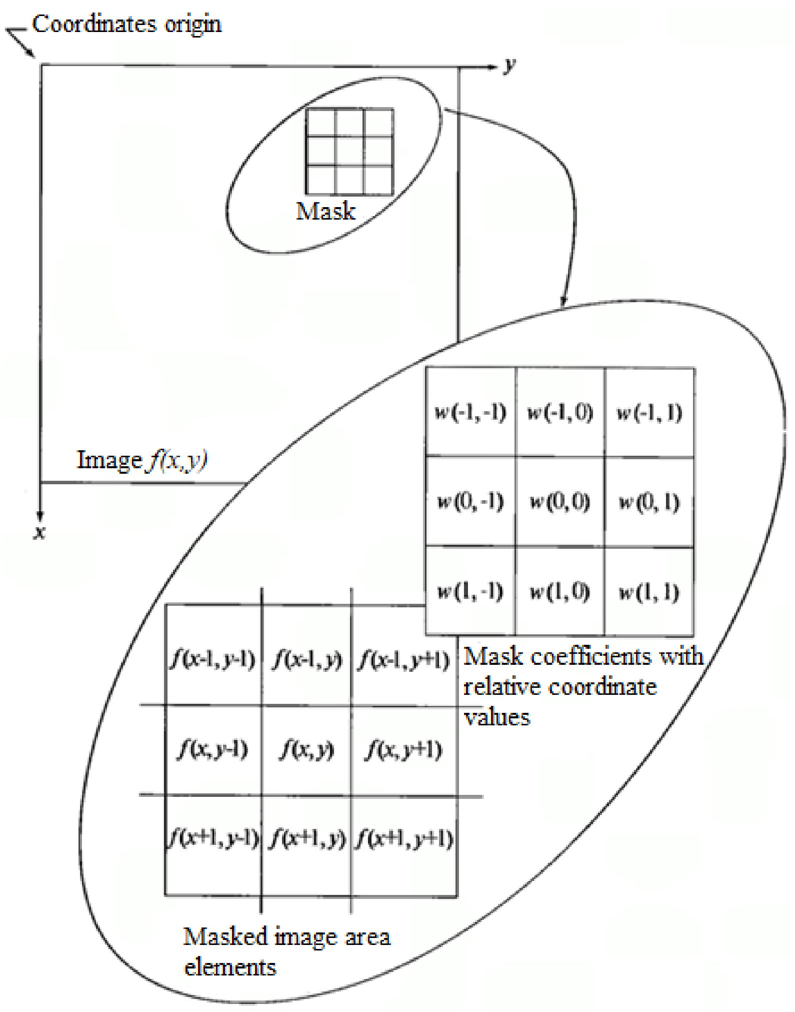


Fig. 3.5. Spatial filtering scheme

The process is based on simply moving the filter mask from point to point in the image; at each point *(x, y)*, the filter response is computed using predefined links. In the case of linear spatial filtering, the response is given by the sum of the product of the filter coefficients by the corresponding pixel values in the area covered by the filter mask. For a 3x3 element mask shown in Figure 3.5, the result (response) *R* of linear filtering at the point *(x, y)* of the image will be:

$$R = w(-1,-1)f(x-1,y-1) + w(-1,0)f(x-1,y) + \cdots$$
$$+w(0,0)f(x,y) + \cdots + w(1,0)f(x+1,y) + w(1,1)f(x+1,y+1) \qquad (3.6)$$

which, as you can see, is the sum of the products of the mask coefficients by the pixel values directly under the mask. In particular, note that the coefficient *w (0,0)* is at the value of *f(x, y)*, indicating that the mask is centered at the point *(x, y)*.

Discrete analogs of the derivatives of the first and second order are used when detecting differences in brightness. For simplicity of presentation, one-dimensional derivatives will be considered.

The first derivative of the one-dimensional function *f(x)* is defined as the difference between the values of neighboring elements:

$$\frac{\partial f}{\partial x} = f(x+1) - f(x) \tag{3.7}$$

Here, we used a partial derivative notation in order to preserve the same notation in the case of two variables *f(x, y)*, where we have to deal with partial derivatives along two spatial axes. The use of a partial derivative does not change the essence of the consideration.

Similarly, the second derivative is defined as the difference between adjacent values of the first derivative:

$$\frac{\partial^2 f}{\partial^2 x} = f(x+1) + f(x-1) - 2f(x) \tag{3.8}$$

The calculation of the first derivative of a digital image is based on various discrete approximations of a two-dimensional gradient. By definition, the gradient of the image *f(x, y)* at the point *(x, y)* is the vector:

$$\nabla f = \begin{bmatrix} Gx \\ Gy \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{3.9}$$

As it is known, the direction of the gradient vector coincides with the direction of the maximum rate of change of the function *f* at the point *(x, y)*.

An important role in the detection of contours is played by the modulus of this vector, which is denoted by ∇f and is equal to

$$\nabla f = |\nabla f| = \sqrt{G_x^2 + G_y^2} \tag{3.10}$$

This value is equal to the value of the maximum rate of change of the function *f* at the point *(x, y)*, and the maximum is reached in the direction of the vector ∇f. The value ∇f is also often called the gradient.

The direction of the gradient vector is also an important characteristic. Let *α(x,y)* denote the angle between the direction of the vector ∇f at the point *(x,y)* and the *x*-axis. As it is known from mathematical analysis,

$$\alpha(x,y) = arctg\left(\frac{Gy}{Gx}\right) \tag{3.11}$$

From here it is easy to find the direction of the contour at the point (x, y), which is perpendicular to the direction of the gradient vector at this point. And you can calculate the gradient of the image by calculating the values of the partial derivatives *∂f/∂x* and *∂f ∂y* for each point.

Let the 3x3 area shown in the figure below (see Fig. 3.6) represent the brightness values in the vicinity of some image element.

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

Fig. 3.6 Neighborhood 3x3 in the image

The use of such a mask by the Prewitt operator is specified by the following expressions:

$$G_x = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3) \tag{3.12}$$

and

$$G_y = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7) \tag{3.13}$$

In these formulas, the difference between the sums along the top and bottom rows of the 3x3 neighborhood is the approximate value of the derivative along the *x*-axis, and the difference between the sums along the first and last columns of this neighborhood is the derivative along the *y*-axis. To implement these formulas, an operator described by the masks in Fig. 3.7 is used, which is called the Prewitt operator.

-1	-1	-1
0	0	0
1	1	1

-1	0	1
-1	0	1
-1	0	1

Fig. 3.7 Prewitt operator masks

3.4. Image binarization using Otsu method

Image binarization is a transformation, the essence of which is that the brightest and most significant pixels for any subsequent processing become as bright as possible, turning into white points (the color with maximum intensity or brightness), and all other points that are considered background, become minimally bright, that is, they are converted to black points (absolute absence of color, minimum brightness or intensity). Thus, the entire binarization operation is reduced to the usual pixel-by-pixel transformation of each point of the image either to white or black, depending on a certain brightness feature, that is, on a certain minimum acceptable brightness value, exceeding which the point becomes white. This feature will be called the binarization threshold, and this is the first thing that needs to be determined when implementing image binarization [12].

At the moment, there are a huge variety of binarization algorithms and methods, ranging from a simple manual one (the threshold is set manually and depending on the image itself) to complex adaptive and multi-methods (including multilayer binarization), but here will be considered an interesting and effective method, which is called the Otsu method.

Otsu's method is an algorithm that allows you to divide image pixels into two classes ("useful" and "background"), due to a simple statistical analysis of the image, which, when dividing pixels into classes, makes sure that the variance within one class is minimal.

Otsu's method looks for a threshold that reduces the variance within a class, which is defined as the weighted sum of the variances of two classes [13]:

$$\sigma_\omega^2(t) = \omega_1(t)\sigma_1^2(t) + \omega_2(t)\sigma_2^2(t) \tag{3.14}$$

where the weights ω_i — are the probabilities of two classes separated by a threshold t, μ_i — the variance of these classes.

Otsu showed that minimizing variance within a class is equivalent to maximizing variance between classes:

$$\sigma_b^2(t) = \sigma^2 - \sigma_\omega^2(t) = \omega_1(t)\omega_2(t)[\mu_1(t) - \mu_2(t)]^2 \qquad (3.15)$$

which is expressed in terms of the probability ω_i and the arithmetic mean μ_i, which, in turn, can be updated iteratively. This idea led to an efficient algorithm:

Let a monochrome image be given $G(i,j), i = \overline{1, Height}, j = \overline{1, Width}$, repetition counter k=0.

1. Calculate the histogram $p(l)$ of the image and the frequency $N(l)$ for each intensity level of the image G.
2. Calculate the initial values for $\omega_1(0)$, $\omega_2(0)$ и $\mu_1(0)$, $\mu_2(0)$.
3. For each value $t = \overline{1, max(G)}$ - semitones - horizontal axis of the histogram:
    1. Updating ω_1, ω_2 and μ_1, μ_2
    2. Calculate $\sigma_b^2(t) = \omega_1(t)\omega_2(t)[\mu_1(t) - \mu_2(t)]^2$
    3. If $\sigma_b^2(t)$ is greater than the existing one, then remember σ_b^2 and the value of the threshold *t*.
4. The desired threshold corresponds to the maximum $\sigma_b^2(t)$

$$N_T = \sum_{i=0}^{\max(G)} p(i),$$

$$\omega_1(t) = \frac{\sum_{i=0}^{t-1} p(i)}{N_T} = \sum_{i=0}^{t-1} N(i), \quad \omega_2(t) = 1 - \omega_1(t),$$

$$\mu_T = \frac{\sum_{i=0}^{\max(G)} i * p(i)}{N_T} = \sum_{i=0}^{\max(G)} i * N(i),$$

$$\mu_1(t) = \frac{\sum_{i=0}^{t-1} i * p(i)}{N_T * \omega_1(t)} = \frac{\sum_{i=0}^{t-1} i * N(i)}{\omega_1(t)}, \ \mu_2(t) = \frac{\mu_T - \ \mu_1(t) * \omega_1(t)}{\omega_2(t)}.$$

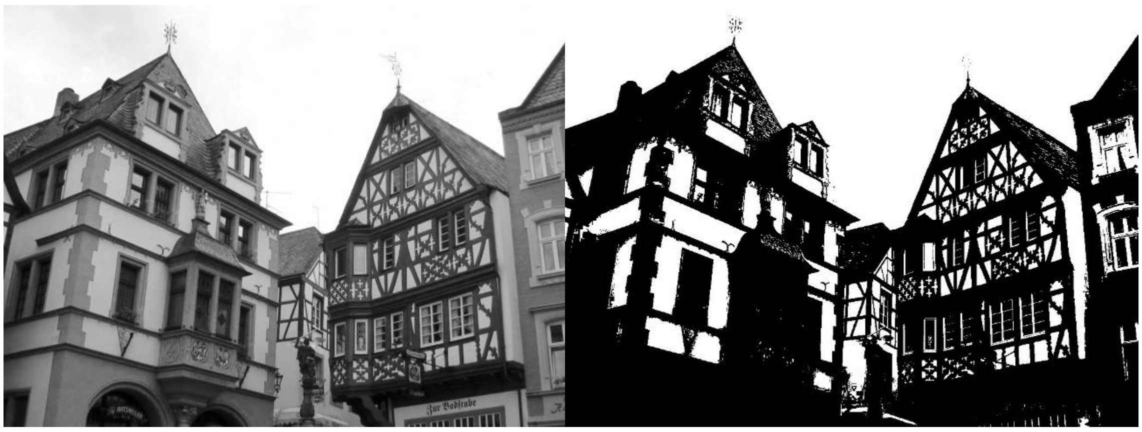


Fig. 3.8. Original image and after binarization with Otsu threshold

3.5. Template matching method description

In order to determine which characters are located on the number plate, the method of comparing the input image with the template, ie the Template matching method, was used [6].

The "similarity" of an image is defined by a certain metric. That is, the pattern is "superimposed" on the image, and the discrepancy between the image and the pattern is considered. The position of the template at which this discrepancy will be minimal, and will mean the location of the desired object.

As a metric, you can use different options, for example, the sum of squared differences (SSD), or cross-correlation (CCORR). Let *f* and *g* be an image and a template with sizes *(k, l)* and *(m, n)*, respectively (the color channels will be ignored); *i, j* - position on the image to which we "attached" the template.

$$SSD_{i.j} = \sum_{a=0..m,b=0..n} \left(f_{i+a,j+b} - g_{a.b}\right)^2 \qquad (3.16)$$

$$CCORR_{i.j} = \sum_{a=0..m,b=0..n} \left(f_{i+a,j+b} - g_{a.b}\right)^2 \qquad (3.17)$$

Let's try to apply the difference of squares to find a kitten on the picture.

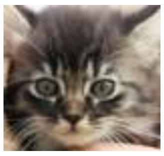


Fig. 3.9. Tamplate image

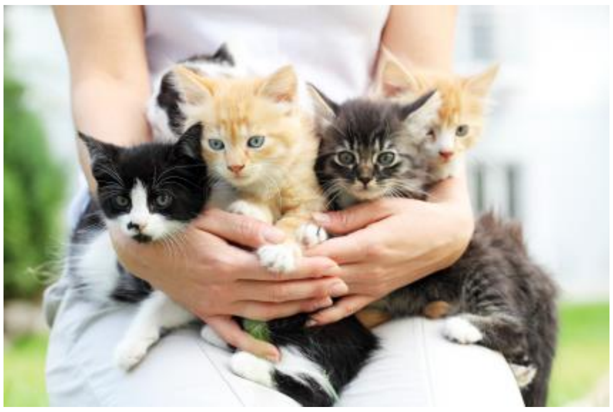


Fig. 3.10. Picture taken from the resource PETA Caring for Cats [6]

On Fig. 3.11 the values of the metric of the similarity of the place in the picture to the template (i.e. *SSD* values for different *i*, *j*). The dark area is where the difference is minimal. This is the pointer to the place that most resembles the template - in the Fig. 3.12. picture this place is circled.

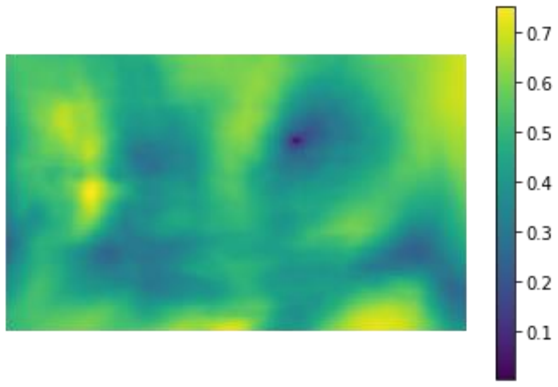


Fig. 3.11. *SSD* values for different *i*, *j*

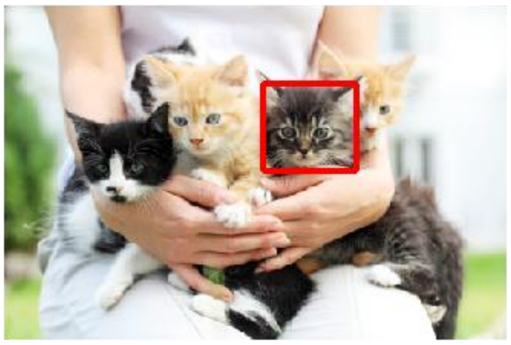


Fig. 3.12. Highlighted matched area

Cross-correlation is actually a convolution of two images. Convolutions can be implemented quickly using Fast Fourier Transform. According to the convolution theorem, after the Fourier transform, the convolution turns into a simple element-wise multiplication:

$$CCORR_{i.j} = f * g = IFFT\big(FFT(f * g)\big) = IFFT\big(FFT(f) \cdot FFT(g)\big) \quad (3.18)$$

Where * is the convolution operator. This way the cross-correlation can be quickly calculated. This gives the overall complexity *O(kllog (kl) + mnlog (mn)),* versus *O(klmn)* when implemented straight. The squared difference can also be implemented using convolution, since after expanding the brackets, it turns into the difference between the sum of the squares of the image pixel values and the cross-correlation:

$$SSD_{i.j} = \sum_{a=0..m,b=0..n} \left(f_{i+a,j+b} - g_{a.b}\right)^2 = \tag{3.19}$$

$$\sum_{a=0..m,b=0..n} f_{i+a,j+b}^2 - 2f_{i+a,j+b}g_{a.b} + g_{a,b}^2$$

$$= \sum_{a=0..m,b=0..n} f_{i+a,j+b}^2 + g_{a,b}^2 - 2CCOR_{i.j}$$

In the case of resizing, the method may not work correctly. This is due to the fact that the method assumes that the object is resized by the same number of times both horizontally and vertically. However, this is not always the case. When the size is changed too much, the distortion caused by the log-to-polar conversion makes the search unstable.

CHAPTER 4. SOLUTION OF NUMBER PLATE DETECTION PROBLEM BY THE DEVELOPED PROGRAM AND EXPERIMENTS

4.1. Description of the experimental setup

In order to make the detection of number plate contour possible such experimental installation is introduced:

1) Mobile phone Samsung galaxy S8 (or WEB-cam).
2) Personal computer with installed Matlab2021.
3) Developed program for computing the algorithm.

To obtain photos, images and videos, was used the optical sensor of the Samsung Galaxy S8 mobile phone, Sony IMX333 with an optical stabilizer, which has a size of 1 / 2.55 ", a resolution of 12.2 MP and a pixel width of 1.4 microns.

Own laptop Dell Precision 7520 was used as a computer. The characteristics of this PC are described in Table 4.1.

Table 4.1. PC characteristics.

PC name	Dell Precision 7520
CPU	Intel Core i7-6820HQ CPU 2,7 GHz
Number of cores	4
Logical processors number	8
Installed RAM	32 Гб
Video adapters	Intel(R) HD Graphics 530 NVIDIA Quadro M2200 4Gb

To develop the program, the Matlab2021 programming environment and its components and libraries Image Acquisition Toolbox, Image Processing Toolbox, Computer Vision Toolbox, Support Package for USB Webcams were installed on the PC. This software is the most accessible and simple to perform the task and allows to easily work with image and video processing.

Below is shown a compact block diagram of the experimental setup.

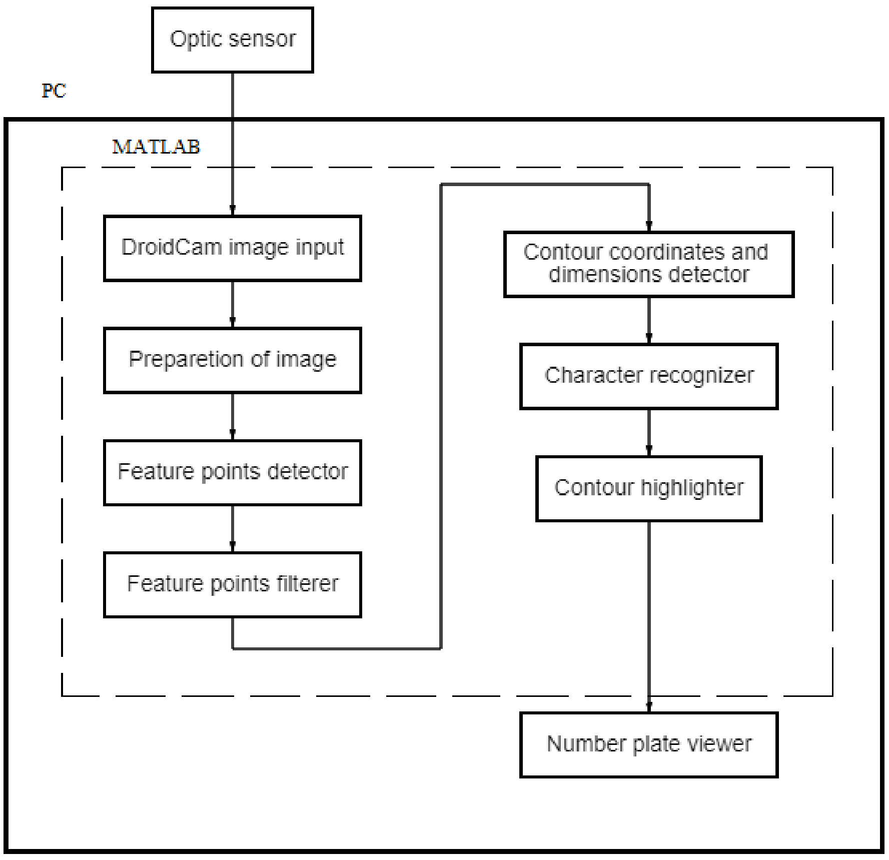


Fig. 4.1. Block diagram of the experimental setup

4.2. Description of the algorithm of the program

Algorithm of work:

1. Recording image data to a computer;
2. Preparation of image data for processing:
    a. Uploading an image to the MATLAB programming environment;
    b. Translate the image into grayscale;
    c. Extra image pixels crop;
    d. Bluring image pixels by Gauss filter;
3. Identification of all characteristic points in the image;
4. Filtering of the found characteristic points;
5. Determining the coordinates and dimensions of the license plate;
6. Recognition of symbols located on the license plate;
7. Selecting the contour of the plate on the input image and output the recognized characters.

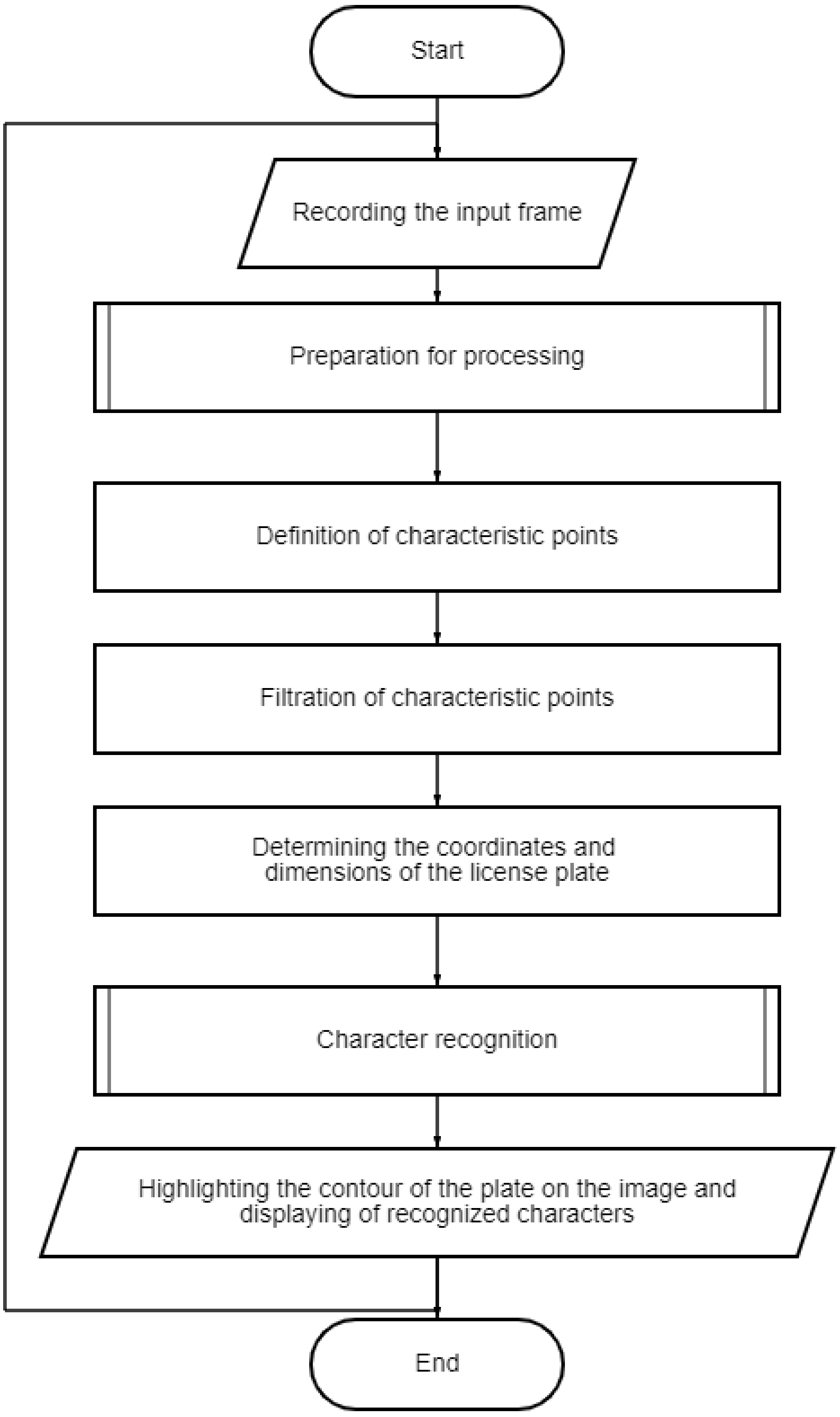
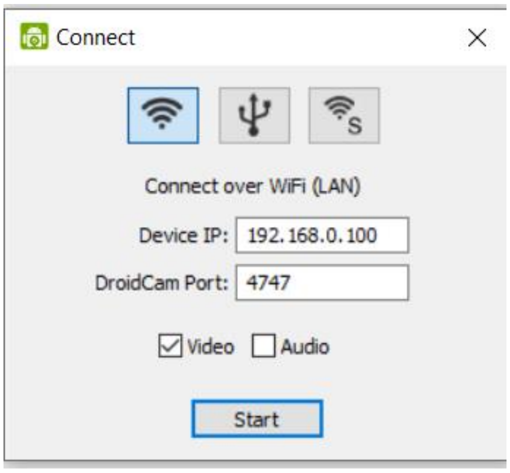

Start
Recording the input frame
Preparation for processing
Definition of characteristic points
Filtration of characteristic points
Determining the coordinates and dimensions of the license plate
Character recognition
Highlighting the contour of the plate on the image and displaying of recognized characters
End

Fig. 4.2. Block diagram of the algorithm of the program work

4.2.1. Recording image data to a computer

To transfer images from the phone's optical sensor to a computer, the phone connects to a computer via DroidCamApp. This program allows you to connect an optical sensor from the phone and transfer data via USB port or Wifi channel (Fig. 4.3 and Fig. 4.4). Both types of connections were used during the experiments.



Device IP: 192.168.0.100
DroidCam Port: 4747
[✓ Video] Audio

Fig. 4.3. Connection of a smartphone camera via DroidCamApp

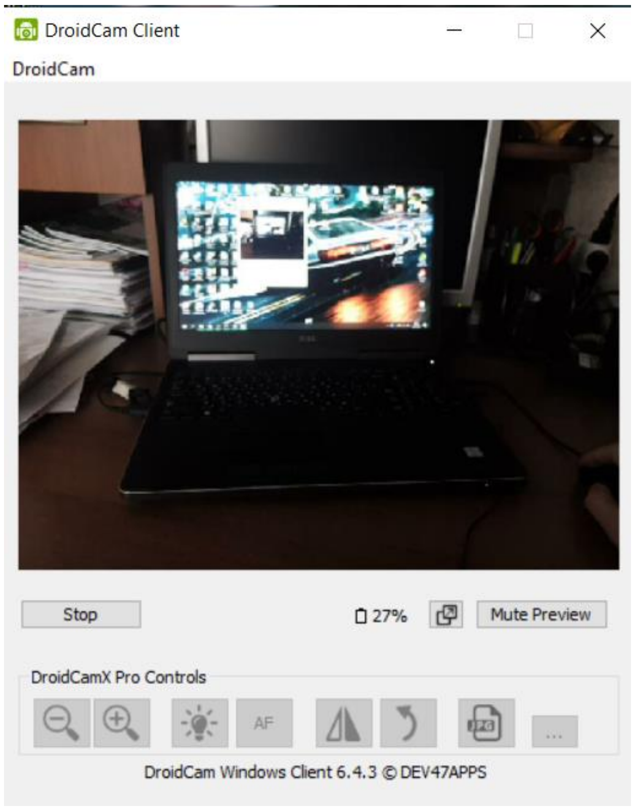


Fig. 4.4. Received video stream from DroidCamApp

Video taken from an optical sensor at 30 frames per second frequency is stored on computer's hard drive. However, for the correct operation of the program it is enough to process every 10th frame, and to demonstrate the algorithm will be used static images with different cars parked in the parking lot.

4.2.2. Preparation of image data for processing

Images from the optical sensor are obtained in RGB format, in a resolution of 1280 × 960 pixels, with a lot of noise and unnecessary information. In order for the data encoded in the image to be processed and various mathematical operations can be performed on it, it must be prepared in advance. This will be described in the following subsections.

a. Uploading an image to the MATLAB programming environment

The first step in image processing is to import it from the drive into the Matlab2021 programming environment, which has the appropriate built-in features and tools.

After importing the image (using the *imread* function), it is written to a variable as a three-dimensional data array, which consists of the values of each pixel on three color channels (red, green, blue), which overlap each other to give the color of the pixel (Fig. 4.5).

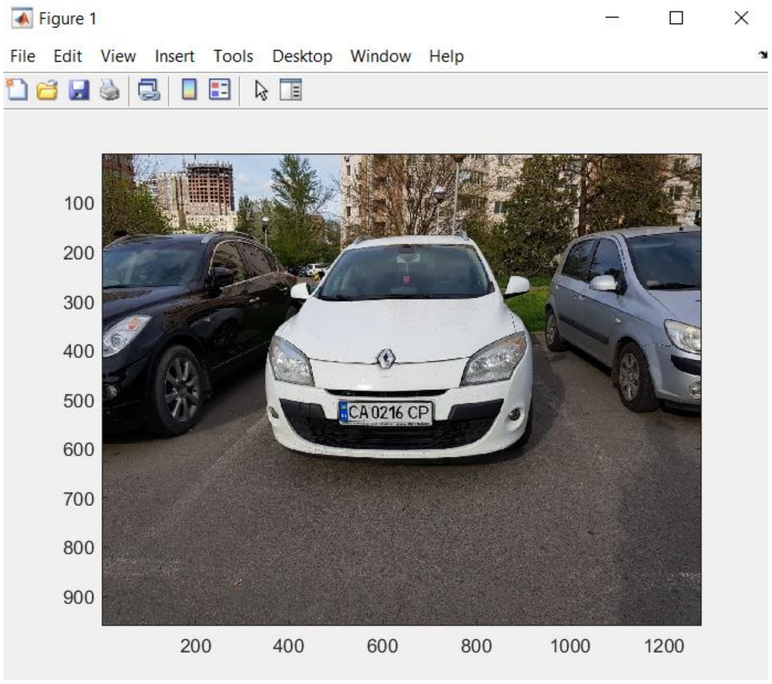


Fig. 4.5. Example of a "raw" downloaded image

Now with these arrays it is possible to carry out calculations in the programming environment.

b. Translate the image into grayscale

Further calculations will be performed with the *uint8* data type, which is an integer from 0 to 255 and takes up 1 byte of memory.

Each of the R, G, B channels may have a pixel value from 0 to 255. The value of their sum can exceed 255, so these three channels must be combined into one grascaled. This problem is solved by means of function *rgb2gray()*.

Function *rgb2gray* converts RGB values to grayscale values by forming a weighted sum of the *R*, *G*, and *B* components:

$$I_{gray} = 0.2989 * R + 0.587 * G + 0.114 * B \tag{4.1}$$

At the output a grayscale image (Fig. 4.6) recorded as a one-dimensional array of points was gotten.

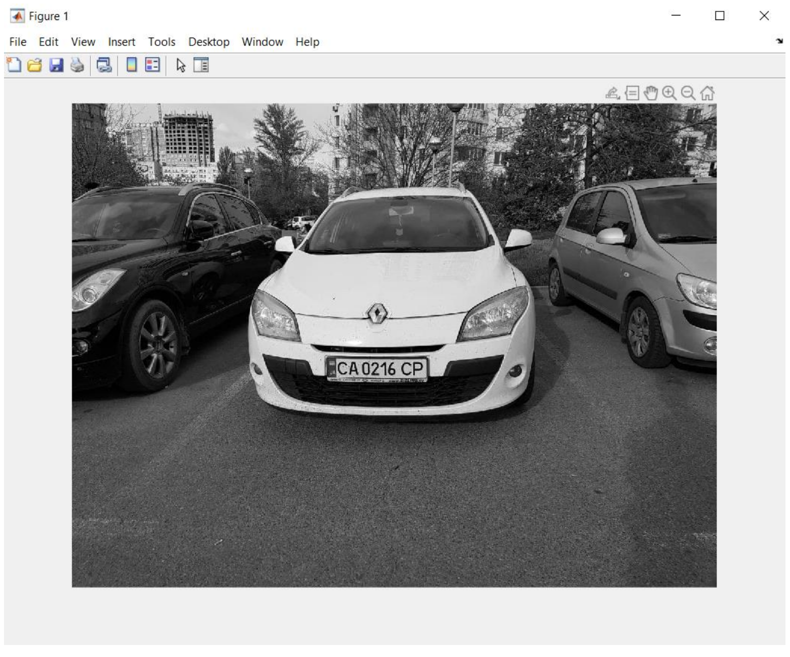


Fig. 4.6. Converted into grayscale image

c. Extra image pixels crop

The picture above shows that the optical sensor captured many extra objects besides the car itself. This will interfere with the identification of characteristic points in the further course of work. To reduce the amount of unnecessary information, the image must be cropped at its edges to a certain number of pixels using the function *imcrop* (Fig. 4.7.).

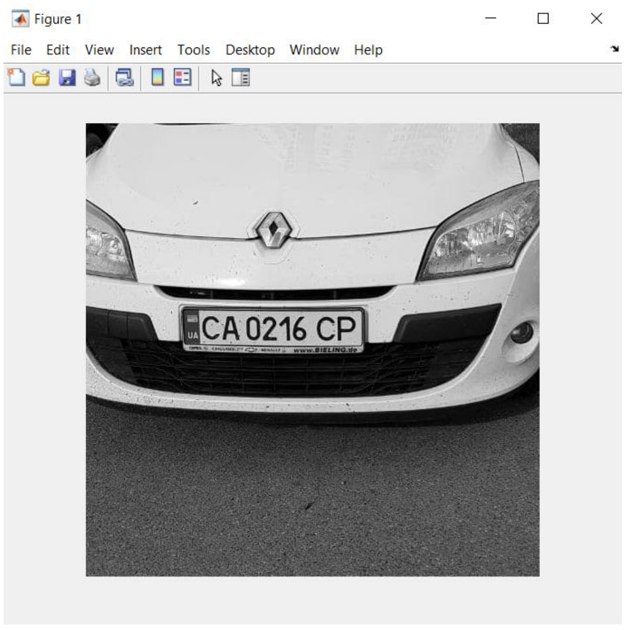


Fig. 4.7. Cropped image

After cropping, the number of pixels and unnecessary information in the image decreased, which has a positive effect on the speed of the program work.

d. Bluring image pixels by Gauss filter

Now, when the cropped grayscale image is done it is needed to smooth unnecessary noises by 2-D Gaussian filtering [14].

Pixels in the sliding window that are closer to the analyzed pixel should have a greater influence on the filtering result than the extreme ones. Therefore, the coefficients of the mask weights can be described by a bell-shaped Gaussian function. When filtering images, a two-dimensional Gaussian filter is used:

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}} * \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{y^2}{2\sigma^2}} \qquad (4.2)$$

The larger is the parameter σ, the more the image is blurred. Typically, the filter radius is $r = 3\sigma$. In this case, the size of the mask $2r$+1×$2r$+1 and the size of the matrix 6σ+1×6σ+1. Outside this neighborhood, the values of the Gaussian function will be negligible. In MATLAB, Gaussian filtering of an image can be performed using the imgaussfilt () function, shown in Fig. 4.8.

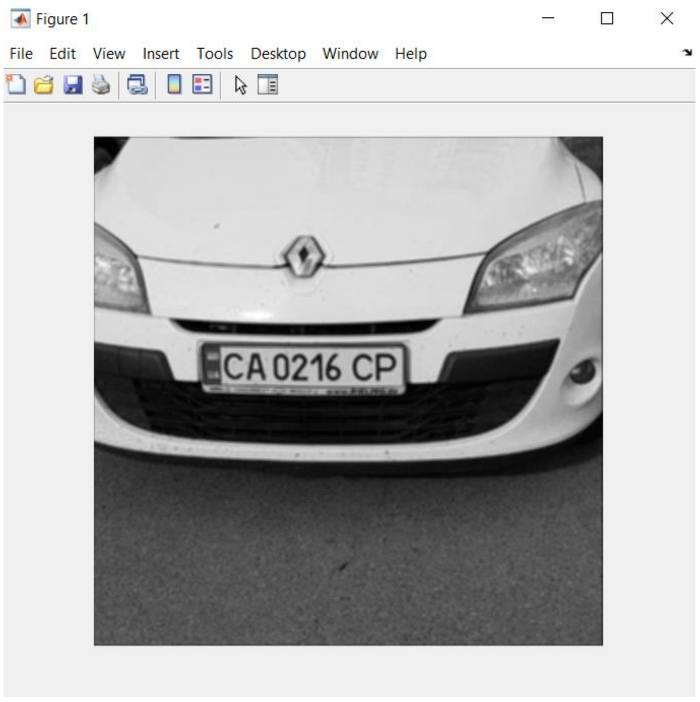


Fig. 4.8. Result of Gaussian filtering

4.2.3. Identification of all characteristic points in the image

The task of the program is to monitor the contour of the target. The tracking object in this case is a car, and the license plate is a common and most distinctive feature which each regular car has. Therefore, its very position will be detecded.

In order to find out the location of the license plate, firstly it is in need to find all the characteristic points in the image, and then process them.

Feature point *m* is a point of the image, the neighborhood of which *o(m)* can be distinguished from the neighborhood of any other point of the image *o(n)* in some other neighborhood of the feature point $o_2(m)$. The process of identifying special points is achieved through the use of a detector and a descriptor.

A detector is a method of extracting specific points from an image. The detector ensures the invariance of finding the same feature points with respect to image transformations. There are many types of characteristic point detectors, the most famous of which are Harris corner detector, FAST, SIFT.

By the research method was determined that the FAST detector is the best for determining the characteristic points. As the name suggests, this algorithm works faster than existing competitors but also fulfills its direct purpose. This algorithm tries to find points that lie at the edges and corners of an object, i.e. in places where contrast drops (Fig. 4.9.).

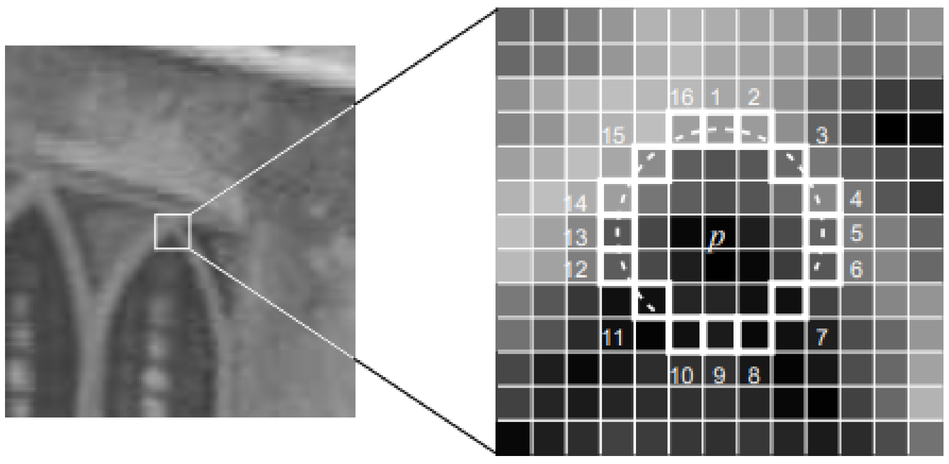


Fig. 4.9. Pixels checked by the algorithm FAST

They are found as follows: FAST builds a circle of radius R around the candidate pixel and checks if there is a continuous segment of pixels of length t on it, which is K units darker (or lighter) than the candidate pixel. If this condition is met, then the pixel is considered a “key point”.

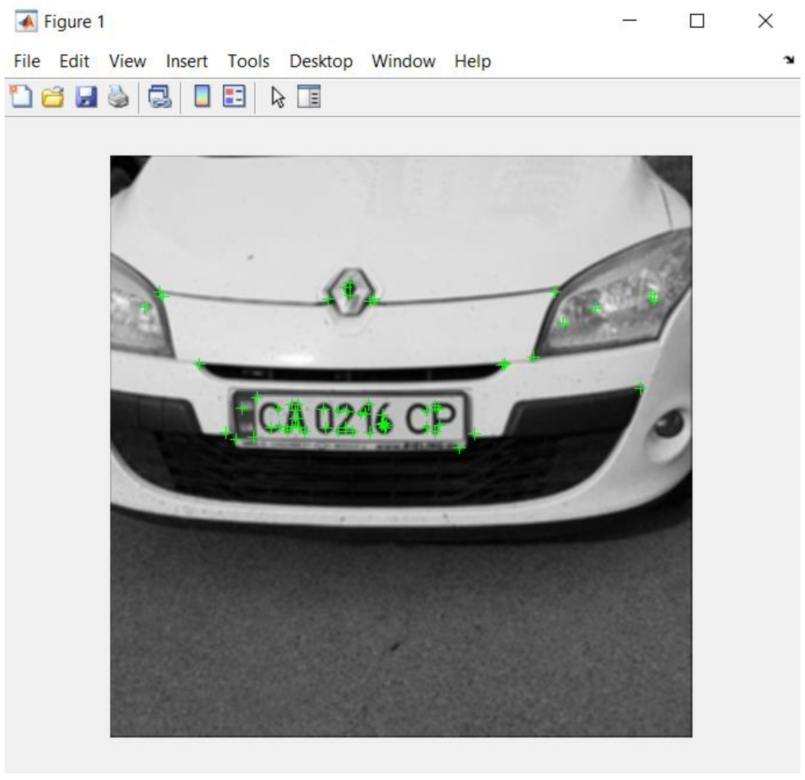


Fig. 4.10. Characteristic points found by the FAST algorithm

4.2.4. Filtering of the found characteristic points

The image above (Fig. 4.10.) shows that the characteristic points were found not only on the license plate itself, but the largest accumulation of them is in this area. This means that they must be filtered under certain conditions.

The filtering algorithm will be based on the knowledge that the largest number of characteristic points is located on the symbols written on the license plate and its edges. Next, the operation of this algorithm will be gradually described.

Firstly write the coordinates *x, y* of each characteristic point in their arrays (Fig. 4.11.) using the function *corners.Location()*.

x	1x60 single
y	1x60 single

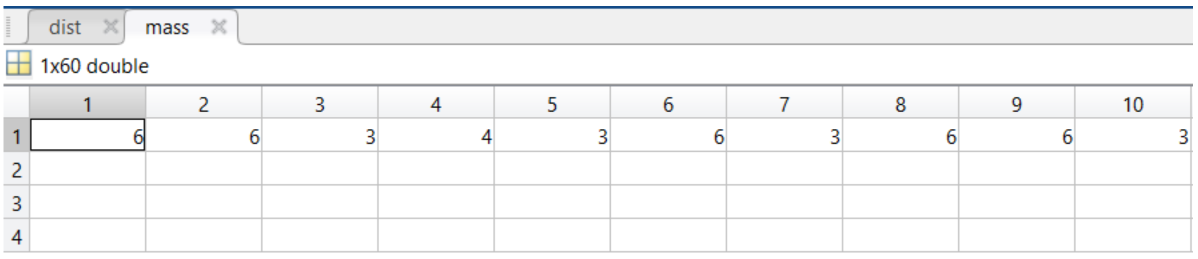
Fig. 4.11. Recorded arrays of coordinates of characteristic points

After the coordinates of the characteristic points are obtained, filter out those that are more than 60 pixels apart from each other. This value was chosen by experiments and proved as optimal. Next, calculate the distance between each point and other characteristic points by the formula:

$$D_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{4.3}$$

Where $D_{i,j}$ – distance between points; x_i, y_i – coordinates of the starting point; x_j, y_j – coordinates of the next point.

Now, for each point whose neighbor is closer than 60 pixels, assign the variable m, which will be equal to the weight coefficient of the point. That is, the more neighbors has a feature point at a distance of up to 60 pixels, the greater will be the value of its weight.



	1	2	3	4	5	6	7	8	9	10
1	6	6	3	4	3	6	3	6	6	3
2										
3										
4										

Fig. 4.12. An array of weight values for each point

When the weight values for each characteristic point are known, they can be filtered by this criterion. For this purpose the minimum threshold of weight of characteristic points m_{min} is entered. The approximate number of characteristic points, which are located on the number plate, is determined experimentally. Based on this, adjust the minimum weight threshold by assigning it a value $m_{min} = 6$, and compare it with the value of the weight of each point. If $m \geq m_{min}$, the point passes the check and its coordinates x_{1i}, y_{1i} are written to a new array.. The coordinates of points that have not passed the check are set to 0.

The last step in filtering points is to remove the coordinates of the points, which now have a value of 0 with the *nonzeros()* function. Now an array of points that fit the desired criterion is found, ie those that are directly on the number plate (Fig. 4.13.).

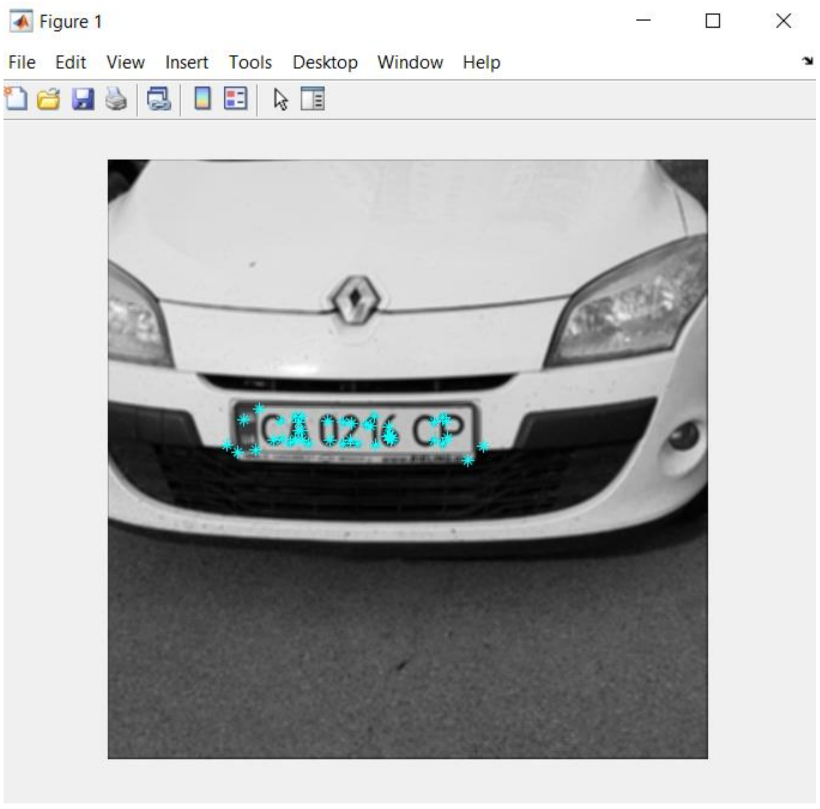


Fig. 4.13. Graphic representation of filtered points

There is also an option that the criterion of the minimum weight of the point may fit smaller clusters of characteristic points, which are further from the number plate but are not far enough to not to pass the distance checking. In this case the following code to determine the coordinates of the number plate will work incorrectly. In this case, m_{min} will increase by 1 until the threshold value of the minimum weight of the point is large enough to weed out the parasitic points, and the program will be executed correctly.

4.2.5. Determining the coordinates and dimensions of the license plate

After filtering the characteristic points, it is safe to say that they all belong only to the license plate, and now the work can proceed to determination of its coordinates and dimensions.

To define the image cropping limits, in order to leave only the image of the plate itself, it is needed to have such data as the coordinates of the upper left corner M_{left} , M_{upper} and the values of the width W and height H of the plate, which can be calculated as follows:

$$W = M_{right} - M_{left} \tag{4.4}$$

$$H = M_{lower} - M_{upper} \tag{4.5}$$

Where M_{left} – the minimum value from the array of coordinates of points x_{1i} (left crop border); M_{right} – the maximum value from the array of coordinates of points x_{1i} (right crop border); M_{lower} – the maximum value from the array of coordinates of points y_{1i} (lower crop border); M_{upper} – the minimum value from the array of coordinates of points y_{1i} (upper crop border).

Knowing the values described above, the image of the license plate is cropped, as shown in Fig. 4.14.

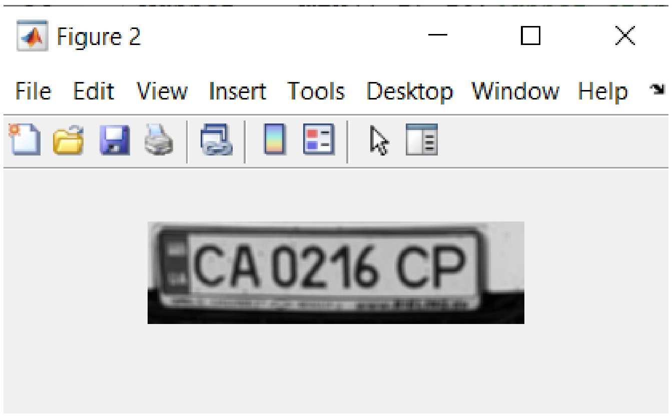


Fig. 4.14. The resulting image of the license plate after cropping according to characteristic points

To further clarifying the coordinates of the number frame, the resulting image must be translated from grayscale to binary.

Image binarization is a transformation, the essence of which is that the brightest and most significant pixels for any subsequent processing become as bright as possible, turning into white points (the color with maximum intensity or brightness), and all other points that are considered to be background, become minimally bright, that is, they are converted to black points (absolute absence of color, minimum brightness or intensity).

In the Matlab programming environment, an image can be binarized using the *imbinarize()* function, which performs it using the Otsu binarization method (Fig. 4.15.).

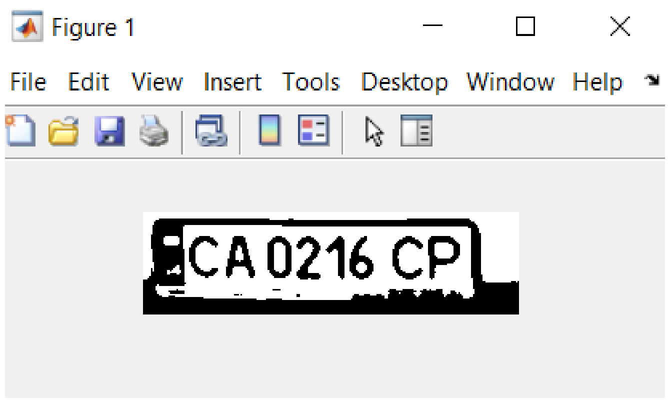


Fig. 4.15. Binary image of the license plate

To further refining of the coordinates, it is needed to select the contours of the objects in the image (Fig. 4.16.). I chose the method of selecting contours using the Prewitt operator.

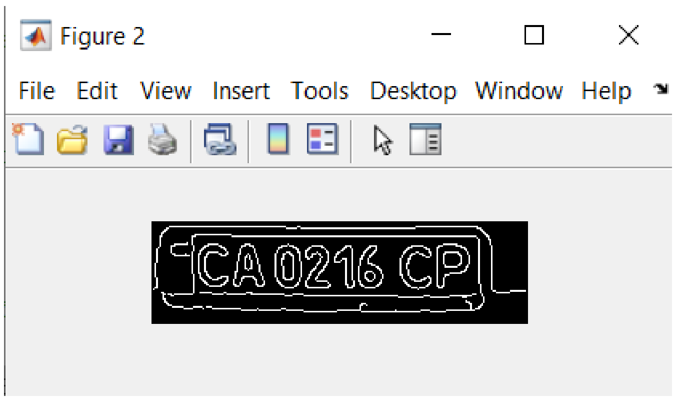


Fig. 4.16. Selected contours in the image

The Prewitt operator is used to select horizontal contours of objects using a mask (Fig. 4.17.).

-1	-1	-1
0	0	0
1	1	1

Fig. 4.17. Horizontal mask of the Prewitt operator

To select vertical contours, this mask is transposed.

Now find the properties of the resulting binary image by the function *regionprops()* for the final cropping of the number plate. These parameters include *Area*, *BoundingBox*, *Image*.

Where *Area* - the actual number of pixels in the closed area; *BoundingBox* - coordinates and dimensions of the smallest rectangle containing a closed area; *Image* - a binary image of the same size as the *BoundingBox* returned as a binary array.

Next, by comparing the *Area* with the *BoundingBox* the coordinates, width and height of the largest BoundingBox are found. This will be the final cropping of the number plate (Fig. 4.18.).

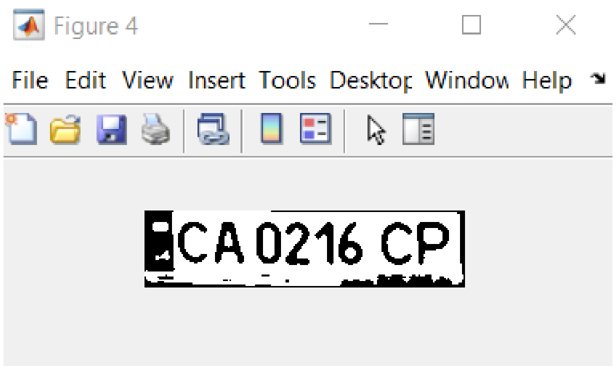


Fig. 4.18. Finally cutted out image of the plate

4.2.6. Recognition of symbols located on the license plate

Since the number plate location and its parameters have already been found, recognizing the characters depicted on it can be started.

Firstly, using the *bwareaopen()* function remove some extra pixels if their closed area is less than 50 pixels, and invert the resulting binary image (Fig. 4.19).

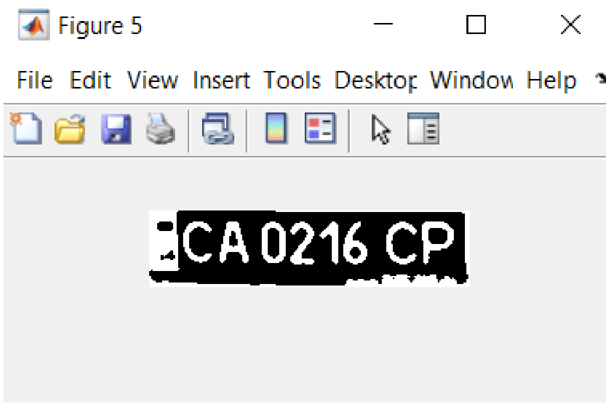


Fig. 4.19. Inverted image with removed noise

Now a kind of preparation for the recognition of letters and numbers is done.

Letter recognition will be based on the Template matching method. This is a method based on finding the place in the image that most closely resembles a template. The "similarity" of an image is defined by a certain metric. That is, the pattern is "superimposed" on the image, and the divergence between the image and the pattern is considered. The position of the template at which this divergence will be minimal, will mean the location of the desired object.

Therefore, to implement this algorithm, firstlly a database of templates for images of letters and numbers was created and saved in folder *Alpha*. Each image is binary, stored in .bmp format and has a resolution of 24 × 42 pixels (see Appendix B).

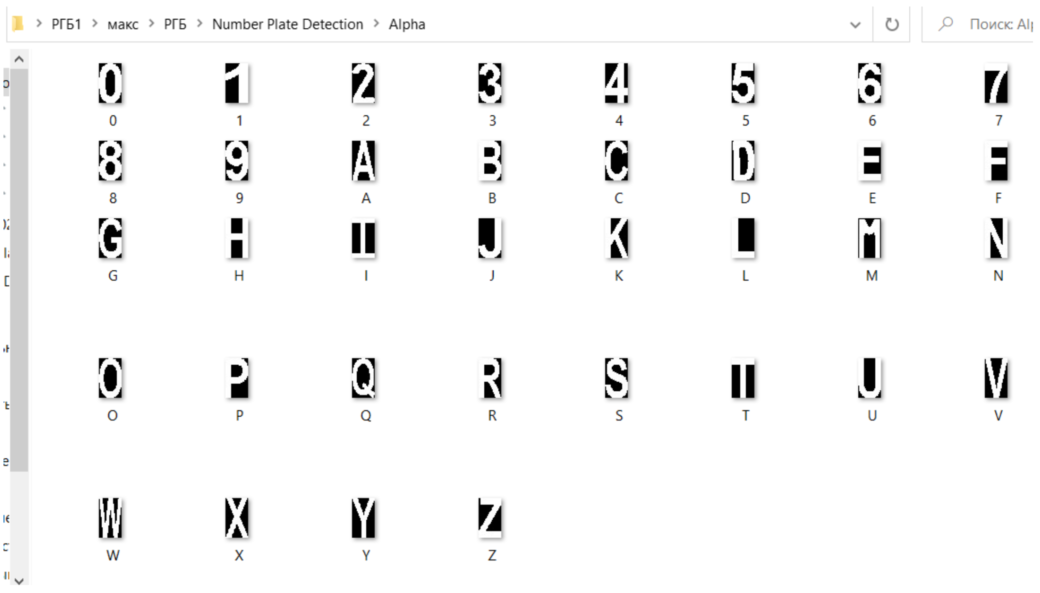


Fig. 4.20. Folder with saved letter and number templates

Each image is written to the appropriate variable, then these variables are written to their arrays *letter* (for letters) and *number* (for numbers), which in turn are entered into a two-dimensional array *NewTemplates*.

After saving the array with images of letters and numbers, the resulting tablet image resolution is changed to be able to compare it with the templates. Each template is compared with the resulting image of the license plate from left to right. For each of them a correlation coefficient is found using the function *corr2()*, which calculates it by the formula:

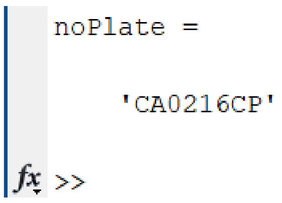
$$r = \frac{\sum_m \sum_n (A_{mn} - \bar{A})(B_{m_n} - \bar{B})}{\sqrt{(\sum_m \sum_n (A_{mn} - \bar{A})^2)(\sum_m \sum_n (B_{mn} - \bar{B})^2)}} \qquad (4.6)$$

Where $\bar{A}$ - arithmetic mean of *A*; $\bar{B}$ - arithmetic mean *B*; *A* – the first input array of points; *B* – the second input array of points; *m, n* – number of points in first and second array.

The larger the is correlation coefficient, the more is the pattern and the input image match (see Appendix C).

Each of the templates has its own assigned index. The *find()* function finds the index with the largest value of the correlation coefficient, and then displays the letter or number that has this index.

Thus there are the read characters from the plate (Fig. 4.21).



```
noPlate =

    'CA0216CP'

>>
```

Fig. 4.21. Recognized symbols from the plate

Sometimes the algorithm can work incorrectly and display more than 8 characters, so there is also a check for the number of characters found. And if their number is greater, then the extra characters are removed.

There are also cases when pre-preparation of the image is not enough and the program finds fewer characters than needed. Therefore, if the number of characters found is less than 8, the algorithm described above is repeated again, and so on until it is executed correctly.

4.2.7. Selecting the contour of the plate on the input image and output the recognized characters

In order for the result of the program to be visible, the number plate must be highlighted. To implement this, the *rectangle()* function will be used, which draws a rectangle by the specified initial coordinates, width and height (Fig. 4.22).

To find these coordinates it is needed to perform inverse calculations, the opposite of those done earlier. So, knowing the number of iterations of the cycle, namely how many times the procedures for cropping the input image was executed, you can easily calculate the coordinates of the number plate frame on the original input image.

And after finding these coordinates, using the function *text()*, an inscription that duplicates the text of the characters that are located on the license plate is displayed (Fig. 4.22.).

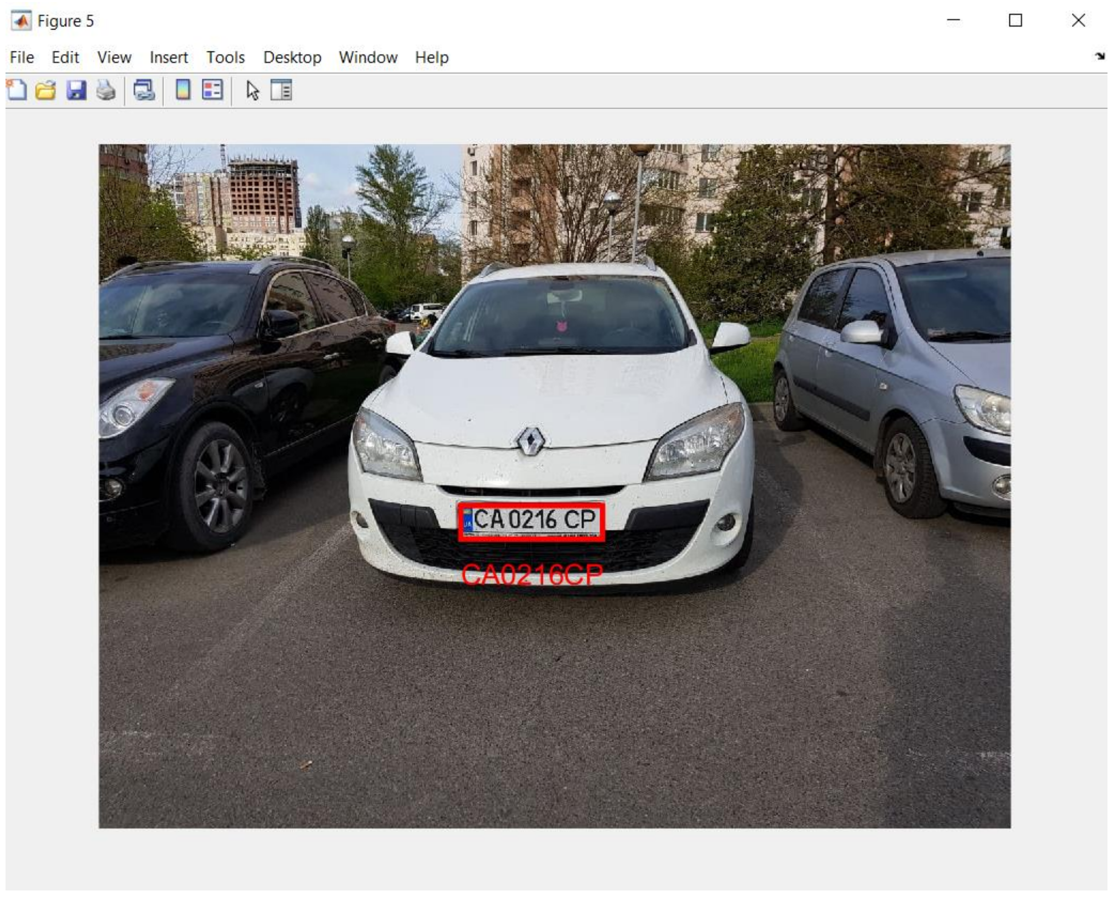


Fig. 4.22. Input image with highlighted number frame and displayed characters

In this way, the program works successfully, tracks license plates and recognizes the symbols on them (The code of the working program is shown in Appendix A).

It, of course, has its limitations, which are manifested in the magnitude of the angle at which the video frame was taken for tracking the contour of the object. Experimentally, it was found that the system finds the coordinates of the number plate correctly until the angle of observation on the vertical and horizontal axes has reached more than about 40 degrees, and the accuracy of detection of symbols on the plate becomes less informative after an angle of 35 degrees.

4.3. Computational load analysis

To use the system effectively, understanding of what computing power it needs is strongly required. After all, if the system loads the computer too much, it will have a bad effect on its speed of work. And the less the system loads the computer, the wider its scope of use can be.

In Table 4.2, how the computer system is loaded when running a single-frame program can be seen. Using the *tic* and *toc* functions from Matlab, the program execution time of the cycle for one frame was calculated. It also can be seen how much RAM was needed for calculations and how much CPU it used. Measurements are presented for 4 different images.

Table 4.2. The results of experiments

Image №	Resolution, pixels	Computing time, sec	Used RAM, Mb	CPU load, %

1	1280×960	0.678983	22	18.5
2	1280×960	0.568815	18	16
3	1280×960	0.577579	19	12.3
4	1280×960	0.535034	19	15

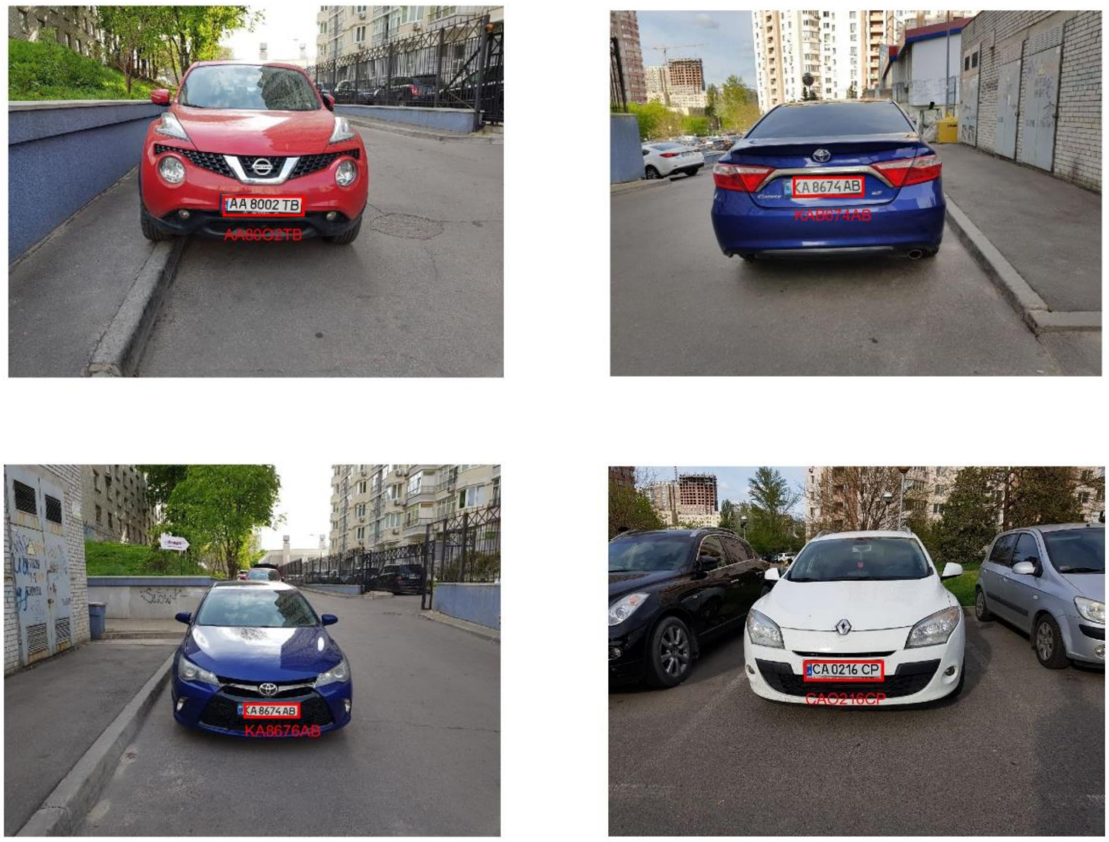


Fig. 4.23. Images used for tests

The idle Matlab programming environment uses approximately 1000 MB of RAM. When running the program, the use of RAM rises to the values specified in the table, but not more than on 30 MB. And when processing the most complex images, the computer's processor is not loaded by more than 20%.

It took from 0.5 to 0.7 seconds to process the image with a resolution of 1280 × 960, which is enough for the normal program work. If there is a need to increase the speed of the program, this can be achieved by reducing the frame resolution. Reducing the resolution by half the processing speed will increase accordingly.

However, such manipulations should be performed carefully, because when the number of pixels in the image are reduced, its informativeness can be lost. Which is not critical for outlining the object of observation, but can greatly affect on the adequacy of the symbols definition on the plate.

So, judging by the experiments on the estimation of computing costs, we can conclude that this system requires up to 2 GB of RAM for stable operation, and does not require large amounts of CPU power. These features allow it to be used in more mobile versions, no longer using a PC for computing, but a Raspberry Pi microcontroller.

CONCLUSION

So, during the implementation of this work, an analysis of existing methods of implementing the algorithm was performed. It was determined that each of them individually cannot fully meet the requirements for the video surveillance system of target contour:

- the correlation detection method is relatively simple, but it is characterized by rather high probability of errors (false detection or missing objects), which is explained by ignoring the properties of noise when synthesizing the image processing algorithm;
- the method of recognizing an object in an image by its contour is not very fast, and can sometimes find erroneous contours;
- the method of recognizing the object in the image by characteristic points does not allow to determine the exact coordinates of the number plate, because the characteristic points for the new license plate will change accordingly;
- Neural network object recognition is quite accurate, but requires relatively large computing resources, the presence of a graphics accelerator and a large number of training sample images.

Therefore, the system's own algorithm based on a combination of methods was developed. There was also developed own algorithm for filtering characteristic points, which always allows to correctly identify the location of the number plate on the image. With the help of Template Matching method has been added function of the character recognition on the license plate.

Compared to existing systems based on deep learning, the developed system also accurately detects the object of tracking, works quickly, and has the ability to be compact using casual webcam and Raspberry Pi microcontroller with installed on MATLAB programming environment.

REFERENCES

1. Системи відеоспостереження та методи виділення контурів на зображеннях / К. Гжешчик та ін. *Управління проектами та розвиток виробництва*. 2018. № 3. С. 79–96.

2. Wang J., Qimei C., De Z., Houjie B. Embedded Wireless Video Surveillance System for Vehicle / International Conference on Telecommunications, Chengdu, China, 2006

3. Что такое компьютерное зрение и где его применяют | РБК Тренды. *РБК Тренды*. URL: https://trends.rbc.ru/trends/industry/5f1f007e9a794756fafbfa83 (дата звернення: 01.06.2021).

4. Компьютерное зрение – Викиконспекты. *Northern Eurasia Contests*. URL: http://neerc.ifmo.ru/wiki/index.php?title=Компьютерное_зрение (дата звернення: 01.06.2021).

5. Сергеев В. В., Гашников М. В. Обнаружение объектов на изображении. С. 10–20.

6. rampeer. Нахождение объектов на картинках. *Все публикации подряд / Хабр*. URL: https://habr.com/ru/company/joom/blog/445354/ (дата звернення: 13.05.2021).

7. lightsource. Детекторы углов. *Все публикации подряд / Хабр*. URL: https://habr.com/ru/post/244541/ (дата звернення: 07.05.2021).

8. Распознавание объектов: 3 вещи, которые необходимо знать. *Сообщество Экспонента*. URL: https://hub.exponenta.ru/post/raspoznavanie-obektov-3-veshchi-kotorye-neobkhodimo-znat244 (дата звернення: 09.05.2021).

9. Фильтр Гаусса (gaussianblurring). *Studbooks*. URL: https://studbooks.net/2016248/informatika/filtr_gaussa_gaussianblurring (да та звернення: 04.05.2021).

10. Unlingator. Детекторы и дескрипторы особых точек FAST, BRIEF, ORB. *Все публикации подряд / Хабр*. URL: https://habr.com/ru/post/414459/ (дата звернення: 14.05.2021).

11. Gepard_vvk. Алгоритмы выделения контуров изображений. *Все публикации подряд / Хабр*. URL: https://habr.com/ru/post/114452/ (дата звернення: 12.05.2021).

12. Бинаризация методом Оцу в dlib – LightHouse Software. *LightHouse Software*. URL: https://lhs-blog.info/programming/dlang/binarizatsiya-metodom-otsu-v-dlib/ (дата звернення: 16.05.2021).

13. Contributors to Wikimedia projects. Метод Оцу – Википедия. *Википедия – свободная энциклопедия*. URL: https://ru.wikipedia.org/wiki/Метод_Оцу (дата звернення: 22.05.2021).

14. Vasylenko M. P., Haida M. V. Video Surveillance System of Target Contour. *Electronics and Control Systems*.

15. Abderrahmane, Ezzahout. Conception and development of a video surveillance system for detecting, tracking and profile analysis of a person / Abderrahmane Ezzahout, Rachid Oulad Haj Thami // 3rd International Symposium ISKO-Maghreb — 2013.

16. Jinsol Ha. Violence detection for video surveillance system using irregular motion information / Jinsol Ha, Jinho Park, Heegwang Kim, Hasil Park, Joonki Paik // 2018 International Conference on Electronics, Information, and Communication (ICEIC) — 2018.

17. Wang J., Qimei C., De Z., Houjie B. Embedded Wireless Video Surveillance System For Vehicle // International Conference on Telecommunications, Chengdu, China, 2006.

18. Beymer, D., McLauchlan, P., Coifman, B., Malik, J. A Real-Time Computer Vision System for Measuring Traffic Parameters // IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1997.

19. Barrenetxea, G., Ingelrest, F., Schaefer, G., Vetterli, M. Wireless Sensor Networks for Environmental Monitoring: The SensorScope experience // IEEE International Zurich Seminar on Communications, Zurich, 2008.

20. Barrenetxea, G., Ingelrest, F., Schaefer, G., Vetterli, M. Wireless Sensor Networks for Environmental Monitoring: The SensorScope experience // IEEE International Zurich Seminar on Communications, Zurich, 2008.

APPENDIXES

Appendix A. Code of the program for highlighting the number plate contour and displaying the characters.

```
close all;
clear all;
tic
im =
imread('C:\Users\Admin\Desktop\РГБ1\макс\РГБ\gg3.jpg');

 im3 = im; % save variable with input image
imgray = rgb2gray(im); % conversion of the image to gray
halftones
imgray = imcrop(imgray, [400 300 500 500]); % crop image
pixels

G = fspecial('gaussian',[4 4],2); % Blur image with
Gaussian filter
 %# Filter it
imgray = imfilter(imgray,G,'same');

corners = detectFASTFeatures(imgray); % determination of
characteristic points by the FAST detector
% imshow(imgray); hold on;
% plot(corners.selectStrongest(200)); % Display points with
selected mass
% pause(2);


for i=1:length(corners.Location) % Record the coordinates
of x, at each characteristic point
    x(i)= corners.Location(i);
    y(i)= corners.Location(i,2);
end
for i=1:length(corners.Location) % Calculation of the mass
of each characteristic point for the distance between them
    mass(i)=0;
    for j=1:length(corners.Location)
    dist(i,j)=sqrt((x(i)-x(j))^2+(y(i)-y(j))^2); % equation
for determining the distance
    if  dist(i,j)<=60 % comparison of the distance between
points with a certain threshold
        mass(i)= mass(i)+1;
    end
    end
end
```

```
masso=6;   % Record the initial threshold of the mass of
points
num_of_iter=1; % initialization of the cycle iteration
count variable

while(1) % Start of the number plate definition cycle and
the symbols on it

for i=1:length(corners.Location) % Cycle of elimination of
characteristic points with too little mass
    if mass(i)<=masso
        mass(i)=0;
    else
        x_1(i)=x(i); % coordinates of points that have been
checked
        y_1(i)=y(i);
    end
end
x_1=nonzeros(x_1)'; % screening points with 0 coordinates
y_1=nonzeros(y_1)';
% imshow(imgray); hold on;
% plot(x_1,y_1,'c*'); % Display of characteristic points
that have passed all checks

Mleft = min(x_1)-0;%left crop edge point
Mright = max(x_1)+20;%right crop edge point
Mupper =  min(y_1)-10;%upper crop edge point
Mlower =  max(y_1)+10;%lower crop edge point

X=400+Mleft*(num_of_iter); % calculation of the trimming
edge after each iteration
Y=300+Mupper*(num_of_iter);

num_of_iter=num_of_iter+1; % cycle iteration counter

imgray = imcrop(imgray, [Mleft Mupper Mright-Mleft Mlower-
Mupper]); % Crop image at extreme characteristic points

imbin = imbinarize(imgray); % image binarization

im = edge(imgray, 'prewitt'); % Select contours on the
cropped image using the Prewitt method
% figure, imshow(im);
% figure, imshow(imbin); % binary cropped image
%Below steps are to find location of number plate
Iprops=regionprops(imbin,'BoundingBox','Area', 'Image'); %
finding image properties: coordinates and dimensions of the
smallest rectangle, actual number of pixels in the region,
binary image of the same size as BoundingBox returned as a
binary array
```

```
area = Iprops.Area; % actual number of pixels in the region
count = numel(Iprops); % number of elements of the array
maxa= area;
boundingBox = Iprops.BoundingBox;

for i=1:count                    % Frame definition
   if maxa<Iprops(i).Area
       maxa=Iprops(i).Area;
       boundingBox=Iprops(i).BoundingBox;
   end
end

im = imcrop(imbin, boundingBox);%crop the number plate area
%figure, imshow(im);
im = bwareaopen(~im, 50); %remove some object if it width
is too long or too small than 50
%figure, imshow(im);
 [h, w] = size(im);%get width and height

%imshow(im);

Iprops=regionprops(im,'BoundingBox','Area', 'Image');
%reading the letter
count = numel(Iprops);
noPlate=[]; % Initializing the variable of number plate
string.

for i=1:count
   ow = length(Iprops(i).Image(1,:)); % image height
   oh = length(Iprops(i).Image(:,1)); % image width
   if ow<(h/2) && oh>(h/3)
       letter=Letter_detection(Iprops(i).Image); % Reading
the letter corresponding the binary image.
       noPlate=[noPlate letter] % Appending every
subsequent character in noPlate variable.
   end
end
masso=masso+1; % increase in the threshold mass of the
points with each cycle
if masso > 13 % Limit the value of the mass of points
    masso_err = 0;
    break
end

if length(noPlate)==9 % Corrects the incorrect definition
of the first character
    if isnumeric(noPlate(3))==0

    noPlate(1)='';
end
```

```
end
if length(noPlate)==8 % output of characters read from the
plate
    masso_err = 1;
    noPlate
    break
end
close all
end
toc
% Frame the area of the number and display the read
characters on
% to the original image
if masso_err == 1
imshow(im3); hold on;
rectangle('Position',[X+boundingBox(1) Y+boundingBox(2)
boundingBox(3)
boundingBox(4)],'EdgeColor','r','LineWidth',3);
text(X+boundingBox(1),Y+boundingBox(2)+2*boundingBox(4),noP
late,'Color','red','FontSize',16);

 end
```

Appendix B. Program code that creates templates of alphabets and numbers.

```
%CREATE TEMPLATES
%Alphabets
A=imread('alpha/A.bmp');B=imread('alpha/B.bmp');C=imread('a
lpha/C.bmp');
D=imread('alpha/D.bmp');E=imread('alpha/E.bmp');F=imread('a
lpha/F.bmp');
G=imread('alpha/G.bmp');H=imread('alpha/H.bmp');I=imread('a
lpha/I.bmp');
J=imread('alpha/J.bmp');K=imread('alpha/K.bmp');L=imread('a
lpha/L.bmp');
M=imread('alpha/M.bmp');N=imread('alpha/N.bmp');O=imread('a
lpha/O.bmp');
P=imread('alpha/P.bmp');Q=imread('alpha/Q.bmp');R=imread('a
lpha/R.bmp');
S=imread('alpha/S.bmp');T=imread('alpha/T.bmp');U=imread('a
lpha/U.bmp');
V=imread('alpha/V.bmp');W=imread('alpha/W.bmp');X=imread('a
lpha/X.bmp');
Y=imread('alpha/Y.bmp');Z=imread('alpha/Z.bmp');

%Natural Numbers
one=imread('alpha/1.bmp');two=imread('alpha/2.bmp');
three=imread('alpha/3.bmp');four=imread('alpha/4.bmp');
five=imread('alpha/5.bmp'); six=imread('alpha/6.bmp');
seven=imread('alpha/7.bmp');eight=imread('alpha/8.bmp');
nine=imread('alpha/9.bmp'); zero=imread('alpha/0.bmp');

%Creating Array for Alphabets
letter=[A B C D E F G H I J K L M N O P Q R S T U V W X Y
Z];
%Creating Array for Numbers
number=[one two three four five six seven eight nine zero];

NewTemplates=[letter number];
save ('NewTemplates','NewTemplates')
clear all
```

Appendix C. Code for comparison of input image with templates.

```
function letter=readLetter(snap)

load NewTemplates
snap=imresize(snap,[42 24]);
rec=[ ];

for n=1:length(NewTemplates)
    cor=corr2(NewTemplates{1,n},snap);
    rec=[rec cor];
end

ind=find(rec==max(rec));
display(ind);

% Alphabets listings.
if ind==1 || ind==2
    letter='A';
elseif ind==3 || ind==4
    letter='B';
elseif ind==5
    letter='C';
elseif ind==6 || ind==7
    letter='D';
elseif ind==8
    letter='E';
elseif ind==9
    letter='F';
elseif ind==10
    letter='G';
elseif ind==11
    letter='H';
elseif ind==12
    letter='I';
elseif ind==13
    letter='J';
elseif ind==14
    letter='K';
elseif ind==15
    letter='L';
elseif ind==16
    letter='M';
elseif ind==17
    letter='N';
elseif ind==18 || ind==19
    letter='O';
elseif ind==20 || ind==21
    letter='P';
elseif ind==22 || ind==23
```

```
    letter='Q';
elseif ind==24 || ind==25
    letter='R';
elseif ind==26
    letter='S';
elseif ind==27
    letter='T';
elseif ind==28
    letter='U';
elseif ind==29
    letter='V';
elseif ind==30
    letter='W';
elseif ind==31
    letter='X';
elseif ind==32
    letter='Y';
elseif ind==33
    letter='Z';
    %*-*-*-*-*
% Numerals listings.
elseif ind==34
    letter='1';
elseif ind==35
    letter='2';
elseif ind==36
    letter='3';
elseif ind==37 || ind==38
    letter='4';
elseif ind==39
    letter='5';
elseif ind==40 || ind==41 || ind==42
    letter='6';
elseif ind==43
    letter='7';
elseif ind==44 || ind==45
    letter='8';
elseif ind==46 || ind==47 || ind==48
    letter='9';
else
    letter='0';
end
end
```