A NEW APPROACH FOR TEXT RECOGNITION ON A VIDEO CARD

An important task is to develop a computer system that can automatically read text content from images or videos with a complex background. Due to a large number of calculations, it is quite difficult to apply them in real-time. Therefore, the use of parallel and distributed computing in the development of real-time or near real-time systems is relevant. The latter is especially relevant in such areas as automation of video recording of traffic violations, text recognition, machine vision, fingerprint recognition, speech, and more. The paper proposes a new approach to text recognition on a video card. A parallel algorithm for processing a group of images and a video sequence has been developed and tested. Parallelization on the video-core is provided by the OpenCL framework and CUDA technology. Without reducing the generality, the problem of processing images on which there are vehicles, which allowed to obtain text from the license plate. A system was developed that was tested for the processing speed of a group of images and videos while achieving an average processing speed of 207 frames per second. As for the execution time of the parallel algorithm, for 50 images and video in 63 frames, image preprocessing took 0.4 seconds, which is sufficient for real-time or near real-time systems. The maximum acceleration of image processing is obtained up to 8 times, and the video sequence – up to 12. The general tendency to increase the acceleration with increasing dimensionality of the processed image is preserved, which indicates the relevance of parallel calculations in solving the problem.

Key words: CUDA technology, real-time system, machine learning, acceleration factor, Gaussian filter, optical character recognition.

Introduction

One of the very important, but non-trivial tasks of machine learning is the problem of text recognition from the image [1-3]. Due to the large number of calculations it is quite difficult to apply it in real time. However, the calculation can be significantly accelerated by using parallelization, which will allow you to perform operations independently for each image.

The main characteristic of the video is the number of frames per second, that the number of consecutive full-screen images that are displayed every second. On average, the human eye can process 12 frames per second. Most smooth motion cameras record at 30 or 60 frames per second [4]. Each of these photos can be used to find or recognize specific objects, process them for a specific purpose, and perform other tasks.

Assuming that the frame rate in the video is 30, you can extract 1800 images from the minute video. Therefore, the problem of speeding up the processing of a group of images and a video sequence has been developed and tested. Parallelization on the video-core is provided by the OpenCL framework and CUDA technology. Without reducing the generality, the problem of processing images on which there are vehicles, which allowed to obtain text from the license plate. A system was developed that was tested for the processing speed of a group of images and videos while achieving an average processing speed of 207 frames per second. As for the execution time of the parallel algorithm, for 50 images and video in 63 frames, image preprocessing took 0.4 seconds, which is sufficient for real-time or near real-time systems. The maximum acceleration of image processing is obtained up to 8 times, and the video sequence – up to 12. The general tendency to increase the acceleration with increasing dimensionality of the processed image is preserved, which indicates the relevance of parallel calculations in solving the problem.

Key words: CUDA technology, real-time system, machine learning, acceleration factor, Gaussian filter, optical character recognition.

Introduction

One of the very important, but non-trivial tasks of machine learning is the problem of text recognition from the image [1-3]. Due to the large number of calculations it is quite difficult to apply it in real time. However, the calculation can be significantly accelerated by using parallelization, which will allow you to perform operations independently for each image.

The main characteristic of the video is the number of frames per second, that the number of consecutive full-screen images that are displayed every second. On average, the human eye can process 12 frames per second. Most smooth motion cameras record at 30 or 60 frames per second [4]. Each of these photos can be used to find or recognize specific objects, process them for a specific purpose, and perform other tasks.

Assuming that the frame rate in the video is 30, you can extract 1800 images from the minute video. If you analyze a video that lasts more than an hour, the number of images will exceed 100,000. On the CPU, the processing of all these images can take a long time, and therefore there is a problem to speed up this process.

To solve the problem described above, it is proposed to use CUDA technology. Its use allows you to get a great acceleration over time, given the presence of drivers and libraries that allow you to run code on a graphics processor [5].

For optical character recognition, three main methods of image processing are used: morphological filter, low-pass filter and classification algorithm [6]. For this study, a sequence of processing was chosen by adding to the original image of its Top Hat transformation and subtracting from the original image of its Black Hat transformation. This sequence allows you to select the contours of the characters, which allows you to find and further process license plates.
Low-spectrum filters in optical character recognition are used mainly for image blur. With the help of the convolution operation and the lower spectrum filter as the core, the image blur effect is achieved. For blurring use averaging filtration, Gaussian filter, median filtration, bilateral (two-way) filter. Averaging filtering is performed by a convolution operation with a normalization filter. This kernel takes the average of all the pixels below it and replaces the central element. The Gaussian filter uses the Gaussian core.

This is a standard and effective blur method that is often used by default. Also, this filter is highly effective in cleaning the image from Gaussian noise. The median filtering occurs by finding the median of the values under the nucleus and replacing the central element with the found median. Effective in cleaning pulse noise (a common example is noise in analog television) [7]. Bilateral filtering is effective in eliminating noise and keeping edges clear, but this operation is very slow compared to other filters.

Among these options, a Gaussian filter was chosen because of the best ratio of execution speed and blur efficiency. The averaging filter and median filtering blurred the edges too much, which could lead to the elements of the number merging with its edges and, consequently, the loss of information. The bilateral filter relative to the analogues is slower, which can slow down the algorithm.

Various algorithms are available for the classification problem, from classification algorithms (Random Forest, Decision Tree, KNN and others) to neural networks (LaNet5, AlexNet, VGG, ResNet and others) [9-13]. Due to the fact that the paper considers image processing, the classification algorithm – KNN [14] was chosen. This algorithm is implemented in the OpenCV library, which is used for image processing, which allows you to use it with the appropriate data type. Because the test system supports CUDA technology, it was decided to use the integrated module in the OpenCV image processing library itself to work with this technology. This solution allows you to seamlessly run image processing algorithms on the video core.

A set of car license plates from kaggle.com was used for testing [15]. This dataset contains 433 photos of cars with license plates. To test the video stream, video recordings of different lengths were created based on the selected dataset.

The relevance of the topic is due to the fact that the methods of optical character recognition are not sufficiently studied. Also, many tasks that use character recognition require high performance for real-time use. This performance, in turn, can be provided by image processing concurrency, for which parallelization on the video core is best, allowing many simple image operations to be performed simultaneously.

The purpose of this work is to develop an algorithm for the parallel processing of a group of images or a video sequence. At the same time, without reducing the generality, the problem of processing images on which there are vehicles is considered, which will allow us to obtain the text from the license plate.

To achieve this aim, we set the following tasks:
1. Define algorithms that will highlight the desired text in the image.
2. Apply these algorithms to find first the license plate and then the characters in it.
3. Classification of found characters, which will allow us to get the text from the image.
4. Parallelization of the algorithm for processing images of vehicle numbers, and analysis of its efficiency in comparison with the sequential version.

The object of research is the algorithm for selecting and retrieving text from images and its parallel implementation.

The subject of the research is the application of sequential and proposed parallel algorithms for obtaining text from images of vehicle numbers and comparing their efficiency.

The main contribution of this article can be summarized as follows:
1. We have for the first time proposed a computationally efficient approach to the parallel processing of a group of images and a video sequence. The problem of image processing on which there are vehicles, allows to obtain text from the license plate, which is especially relevant for the development of real-time and near real-time systems to automate video recording of traffic violations.
2. We have developed an algorithmic implementation of the proposed approach. The OpenCL framework and CUDA technology were used.
3. We have demonstrated an increase in the processing speed of a group of images and a video sequence. Thus, within the acceleration indices 8 were obtained – for image processing and 12 – for video. We reduced computational complexity using the proposed algorithm. The effectiveness of the proposed approach with increasing the number and dimensionality of the processed images is confirmed.

The rest of this article is organized as follows: literature analysis is presented in Section 2. A detailed description of the sequential and proposed parallel algorithm is given in Section 3. Section 4 contains the results of numerical experiments of algorithmic implementation of the proposed approach, and their discussion is presented in Section 5. Studies are presented in the last section.

Related works

In most cases, OCR (optical character recognition) technology is used to recognize text, which allows you to convert images, PDF files or scanned documents into editable formats. The technology itself consists of two stages: image pre-processing, character recognition for a certain part of the image [16].
An example of the implementation of this technology is the work [17], the purpose of which was to extract text from images to automate the input of nutrition data into fitness applications. The article gives an example of image processing and achieved an accuracy of over 80% for standard font recognition.

Proper pre-processing of the image is of great importance for the success of text recognition. For example, in [18] describes in great detail the preparation of the image for extracting text from it and as a result for different alphabets (Chinese, Arabic and Urdu) was achieved an accuracy from 86 to 99%.

Also, the result of the image processing stages is illustrated by the authors in the work [19] and describes the methods of implementation in C++.

In [20], the K-Nearset Neighbor algorithm was used for character recognition. The OpenCV library was used for both image processing and classification. After analyzing this work, we used the KNN algorithm for character recognition.

An example of using KNN for our model problem, namely license plate recognition, is also described in the article [21]. The difference in the stage of image preparation is the use of the principal components method to remove the features of reducing the dimensionality of the image set.

After processing the source data, it was noticed that there was no parallel processing of the preprocessing stages for a large number of images, as well as a small amount of data on the method of extracting characters (mostly mentions of the Tesseract package). Our work is an attempt to solve this problem.

Methodology

Sequential image processing algorithm
1. Apply morphological transformations to one image.
   A combination of Top Hat and Black Hat operations is used as morphological transformations. Top Hat (Tw) operation is defined as
   \[ T_w(f) = f - f \circ b, \]
   where \( f \) this image is grayscale, \( \circ \) indicates the opening operation, \( b \) is a structural element. A structural element is a binary matrix that defines the structure in units.
   For Black Hat (Tb), the operation will look like this:
   \[ T_b(f) = f \cdot b - f, \]
   where \( \cdot \) is a closing operation.
   The opening operation is defined as:
   \[ A \circ B = (A \ominus B) \oplus B, \]
   where \( A \) is the original set of values, and \( B \) is a structural element; \( \ominus \) — erosion operation, \( \oplus \) — expansion operation.
   Closing is the operation inverse to opening and is defined as follows:
   The erosion operation is mathematically represented as:
   \[ (f \ominus b)(x) = \inf_{y \in E} f(x + y) - b(y), \]
   \( \inf \) is an infimum, \( f(x) \) is an image, \( b(x) \) is a structural element, \( B \) is the space in which \( b(x) \) is defined.
   The expansion operation then looks like this:
   \[ (f \oplus b)(x) = \sup_{y \in E} f(y) + b(x - y), \]
   where \( \sup \) is the supremacy, \( f(x) \) is the image, \( b(x) \) is the structural element, \( E \) is the space in which it is defined.
2. Apply to one image of a Gaussian filter.
   Gaussian blur is an image convolution operation with a Gaussian kernel:
   \[ I_{\sigma} = I \cdot G_{\sigma}, \]
   where \( I \) is the original image, \( \cdot \) — convolution operation, \( G_{\sigma} \) — two-dimensional Gaussian kernel with standard deviation \( \sigma \), \( I_{\sigma} \) — the resulting image after applying the filter with \( \sigma \).
   The Gaussian nucleus with normal distribution is defined as:
   \[ G_{\sigma} = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2 + y^2}{2\sigma^2}}, \]
   where \( x \) and \( y \) are the coordinates of the image value on the respective axes \( OX \) and \( OY \).
   Thus, the two-dimensional Gaussian function will look like this:
   \[ G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2 + y^2}{2\sigma^2}}. \]
3. Repeat steps 1 and 2 for all images.

Parallel image processing algorithm
1. A separate stream is allocated for each image.
2. A parallel algorithm of morphological transformations is applied to all images.
   Morphological transformations are performed by convolution operation with specific operation cores. Unlike a sequential algorithm, where the convolution operation is applied using a floating window, that the value of each element is alternately calculated on the processor core, CUDA technology allows you to calculate the value of each...
image element simultaneously. That is, there is a division of the algorithm by memory, which allows you to perform one operation on different data, which, in turn, corresponds to the purpose of the video core.

Thus, the time complexity of the algorithm will be reduced by simultaneously calculating the resulting convolution values for the image or part thereof.

For every element \((i, j)|i \in m, j \in n\) matrix of the dimension of the nucleus \(B_{wh} \) \(B_{wh}\) where the center of the matrix will be the selected element \((i, j)\) and multiply it by the kernel. The center of the resulting matrix will be an element of the matrix, which will represent the image \(A\) with the applied filter.

3. A parallel filtering algorithm is used for the image.

As with the sequential algorithm, a Gaussian filter with an identical core and function is applied to the image, but thanks to CUDA technology, this operation is performed on the entire image at once. The approach is identical to accelerating the application of morphological transformations, because the Gaussian filter is also used by the convolution operation, only with the Gaussian kernel.

Because GPU cores are not completely independent, but can only perform the same calculations, each action on the matrix becomes equivalent to the action on one matrix element. That is, if we greatly simplify the idea of CUDA architecture, then a matrix of size \(320 \times 2\) when adding \(1\) will perform on each of the \(640\) cores an operation equivalent to adding to the \((i, j)–th\) element of the matrix unit.

Since image processing operations are mainly the use of a convolution operation with a certain core that returns images of the same size, all these actions can be reduced to the use of a filter multiplied by a constant value equal to the number of filters used in the algorithm, and therefore the complexity of the algorithm can be generalized to the following formula:

\[
O(\text{width} \cdot \text{height} \cdot \text{width kernel} \cdot \text{height kernel}).
\]

Accordingly, when paralleling with CUDA, the complexity over time will be equal to the complexity of the algorithm divided by the number of threads:

\[
\frac{O(\text{width} \cdot \text{height} \cdot \text{width kernel} \cdot \text{height kernel})}{N}.
\]

where \(N\) – number of threads.

**Experiments**

The tests were performed on a system with the following characteristics:

- CPU: core i5-8300H
- RAM: DDR4 2667 MHz 24 Gb
- GPU: NVIDIA GeForce GTX 1050
- GPU RAM: GDDR5 4096 MB
- CUDA Cores: 640
- Memory Interface: 128-bit
- GPU Interface: PCI Express x8 Gen3

The results of the software implementation of OCR are recognized text. An example of text recognition together with the stages of image transformation obtained on the basis of the developed information system is presented in Figure 1.

![Fig. 1. Example of the result of OCR software implementation](image-url)
In Table 1 – 4 shows the values of the execution time of the sequential image processing algorithm are presented for each group, the number of measurements – 5. These tables allow us to find the average execution time of the sequential algorithm. A total of 100 tests were performed on the CPU for comparative image processing performance.

### Table 1

**Measurements of the execution time of a sequential algorithm depending on the number of images**

<table>
<thead>
<tr>
<th>Number of images</th>
<th>Origin processing time, ms</th>
<th>Average time, ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>906</td>
<td>896</td>
</tr>
<tr>
<td>100</td>
<td>2564</td>
<td>1915</td>
</tr>
<tr>
<td>433</td>
<td>10463</td>
<td>7990</td>
</tr>
<tr>
<td>1000</td>
<td>18621</td>
<td>19120</td>
</tr>
<tr>
<td>2000</td>
<td>38595</td>
<td>37678</td>
</tr>
<tr>
<td></td>
<td>Plate processing time, ms</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>365</td>
<td>358</td>
</tr>
<tr>
<td>100</td>
<td>740</td>
<td>716</td>
</tr>
<tr>
<td>433</td>
<td>3392</td>
<td>3722</td>
</tr>
<tr>
<td>1000</td>
<td>7709</td>
<td>7901</td>
</tr>
<tr>
<td>2000</td>
<td>16594</td>
<td>16410</td>
</tr>
</tbody>
</table>

From the results of Table 1 you can see that photo processing is much faster for license plates. This is due to the fact that the license plate is a cut part of the overall image and its dimension is much smaller.

### Table 2

**Measurements of the execution time of a sequential algorithm depending on the number of frames in the video**

<table>
<thead>
<tr>
<th>Number of frames</th>
<th>Origin processing time, ms</th>
<th>Average time, ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>63</td>
<td>1301</td>
<td>1359</td>
</tr>
<tr>
<td>150</td>
<td>6681</td>
<td>5991</td>
</tr>
<tr>
<td>300</td>
<td>10277</td>
<td>8619</td>
</tr>
<tr>
<td>600</td>
<td>20774</td>
<td>17662</td>
</tr>
<tr>
<td>1200</td>
<td>40441</td>
<td>34648</td>
</tr>
<tr>
<td></td>
<td>Plate processing time, ms</td>
<td></td>
</tr>
<tr>
<td>63</td>
<td>332</td>
<td>372</td>
</tr>
<tr>
<td>150</td>
<td>1247</td>
<td>1107</td>
</tr>
<tr>
<td>300</td>
<td>3647</td>
<td>3359</td>
</tr>
<tr>
<td>600</td>
<td>6440</td>
<td>6349</td>
</tr>
<tr>
<td>1200</td>
<td>14249</td>
<td>14283</td>
</tr>
</tbody>
</table>

From the results of Table 2 it can be seen that the trend of decreasing processing time for general images and license plate images persists. But because for video, unlike a set of images, each frame has the same resolution, the average execution time increases more smoothly.

In Tables 3 – 4 show the data processing time for the proposed parallel algorithm using GPU and CUDA technology.

### Table 3

**Measurements of execution time of the proposed parallel algorithm for image processing on the video core**

<table>
<thead>
<tr>
<th>Number of frames</th>
<th>Origin processing time, ms</th>
<th>Average time, ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>399</td>
<td>429</td>
</tr>
<tr>
<td>100</td>
<td>506</td>
<td>491</td>
</tr>
<tr>
<td>433</td>
<td>1165</td>
<td>1152</td>
</tr>
<tr>
<td>1000</td>
<td>3151</td>
<td>3373</td>
</tr>
<tr>
<td>2000</td>
<td>5180</td>
<td>5193</td>
</tr>
<tr>
<td></td>
<td>Plate processing time, ms</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>188</td>
<td>184</td>
</tr>
<tr>
<td>100</td>
<td>308</td>
<td>314</td>
</tr>
<tr>
<td>433</td>
<td>1341</td>
<td>1364</td>
</tr>
<tr>
<td>1000</td>
<td>3575</td>
<td>3895</td>
</tr>
<tr>
<td>2000</td>
<td>7076</td>
<td>7046</td>
</tr>
</tbody>
</table>

In Table 3 in comparison with the results of Table 1, it is seen that the average processing speed of the general images increases from 2 to 12 times. This is due to the fact that the GPU-parallel algorithm applies each operation simultaneously to all data. The speed of processing license plate images was initially lower than the processing of the full image, but later it became higher. This is due to the fact that in this case the small dimension of the license plate image does not allow to fully obtain the acceleration of parallel image processing on the GPU and this indicates the reliability of the results.
According to the data presented in Table 4, video stream frame processing with more than 150 frames is performed more efficiently for larger images. Which corresponds to the general trend. The best result for processing license plate images at a video frame length of 63 frames may be due to the fact that the number of images transmitted for processing was small enough that the resources allocated to the image processing stream sufficiently affect the efficiency of the algorithm.

Based on the data obtained from the above tables, the following graphs were compared comparing the results for further analysis.

From Figure 2 shows that with increasing number of images that are processed, the difference in execution time between serial and parallel versions of the algorithm – increases. It is also noticeable that the processing of larger images gives a greater gain in time than the processing of smaller ones.
From the graph (see Figure 3) of the speed comparison it is seen that with increasing number of frames in the video, which is processed, the difference in execution time between consecutive parallel processing options increases. There is also the same trend as in previous tests - processing larger images gives more gain. The processing of smaller images has a less stable trend than in the previous version, which can be caused by different image complexity.

**Fig. 4. Graph comparing the acceleration of image processing on the GPU depending on the number of frames and image size**

This graph (see Figure 4) of image processing acceleration showed acceleration up to 8 times. You can also see a sharp increase in the processing of 433 photos and a small decline in 1000. The main reason for this is the different resolution ability of photos, because the implementation of high-quality image processing takes much longer.

**Fig. 5. Graph comparing the acceleration of video processing on the GPU depending on the number of frames and image size**

From Figure 5 shows that the acceleration in this case reaches 12, and increases much more smoothly. This is because, unlike a set of many photos of vehicles with different resolutions, video frames have an identical extension. Therefore, this graph describes the acceleration for real-time image processing much more accurately.

**Discussion of research results**

The algorithmic implementation of the proposed parallel approach is performed in the work. As a result, a system was developed for which the average image processing speed is 207 frames per second. This speed allows you to build real-time or near-real-time systems for OCR. Such systems allow automating the video recording of violations and omissions in the road sector, which is especially relevant today.

To compare the execution speed of the serial and the proposed parallel image processing algorithm, 5 tests were performed for 5 different image sizes and 5 videos for 2 groups of image sizes for both serial and parallel algorithms. For further analysis, the average results of 5 tests for each of the groups were taken.

Based on the results of testing sequential and parallel image processing algorithms for OCR, a comparison chart of the average acceleration for image processing was constructed (see Figure 6).

According to this graph, we can conclude that when processing video, where a larger type of image (from 600x400) is more efficiently processed on the GPU. Also from this graph it is seen that at small images of parallelization on GPU by means of SUDA technology practically does not give an increase, however still almost 1.5 times more effectively than the consecutive algorithm. Accordingly to the results of numerical experiments, the feasibility of the proposed parallel algorithm based on CUDA technology for GPUs is confirmed. To improve the results, you can transfer the calculation of small images to the CPU, which can achieve much greater acceleration.
**Conclusions**

This paper analyzes image processing algorithms that allow you to select and retrieve text from photos, and identifies 3 main stages: the use of a morphological filter, the use of a blur filter to eliminate noise and the classification of the results.

First, the structure of the morphological filter was studied. The next step was to choose one of the blurring algorithms and the algorithm for classifying the results. The Gaussian filter was chosen as the blur and noise reduction algorithm because it is the highest quality and most effective for most conventional images. Among the classification algorithms, in turn, was chosen KNN because of the simplicity of its implementation.

Next, numerous experiments were conducted, in which there was a comparison of sequential and parallel implementation of the algorithm for processing images of cars. The results showed that the acceleration when processing large images in parallel is much greater than small ones. This is because GPU parallelization applies one operation to the entire image at a time, which is effective for high-resolution photos. In terms of execution time, image reprocessing took 0.9 seconds for 50 images and more than 1.3 seconds for 63-frame videos. Such results are not sufficient for real-time or near real-time systems to automate video recording of violations. In turn, the algorithm paralleled by the GPU showed a time of 0.4 seconds for both cases. In this case, the acceleration is doubled when processing images, and 3 times when processing video. And this value can increase up to 12 times if you increase the number of frames.

Prospects for further research are the selection and comparison of the application of other classification algorithms, as well as the maximum increase in the dimensions of the processed images. Also, the system developed in the work can be implemented as a system of automation of video recordings of traffic violations and passes in the road sector.

**References**

4. FPS (Frames Per Second) Definition - TechTerms. [https://techterms.com/definition/fps](https://techterms.com/definition/fps).


17. Matsuoka Nate and Sullivan Rick. Image processing for the extraction of nutritional information from food labels. Computer Science and Engineering Senior Theses, 42, 2015.


