Journal of Electrical Engineering 6 (2018) 46-51
doi: 10.17265/2328-2223/2018.01.007

# Objects Detection and Recognition System Using Artificial Neural Networks and Drones

Dymitr PIETROW and Jan MATUSZEWSKI
*Military University of Technology, Warsaw 00-908, Poland*

**Abstract:** The paper presents the digital image objects detection and recognition system using artificial neural networks and drones. It contains description based on the example of person identification system where face is the key of object processing. It describes the structure of this system and components of the learning sub-system as well as the processing sub-system (detection, recognition). It consists of the description and examples of learning and processing algorithms and applied technologies. The results of calculations of efficiency and speed of each algorithm are presented in the table and appropriate characteristics. The article also describes the possibilities of further system developments.

**Key words:** Neural networks, detection, recognition, drones.

## 1. Introduction

Image recognition and detection is a complex process that demands a lot of calculations. It requires to use complex decision systems. Their speed depends on the processing algorithms applied and the implementation method of these algorithms. The implementation should efficiently use the hardware platform on which the system runs.

Usage of AI (artificial intelligence) methods and machine learning in processing, allows to gain the greater system effectiveness than using algorithms and rules that were established by programmer at the time of system implementation [2, 3, 10, 11].

By fusing such system with movable sensors which can provide stream of digital images, allows us to observe (in real time) a selected area under search for requested objects. Additionally, if our system has the base data of object types for the requested object, we can classify our object more accurately.

The system proposed in this article is used for people identification who are located in an area of a special security. The system contains the learning sub-system and processing sub-system. The next chapters describe each component of this system and most significant algorithms which have particular influence on the system's performance.

**Corresponding author:** Jan MATUSZEWSKI, Ph.D., lecturer, research field: telecommunication.

## 2. System Structure

### *2.1 Learning Sub-system*

The learning sub-system is a software platform consisting of two applications. The first application is used for generation patterns needed for learning process, whereas the second application teaches our system detecting and recognizing requested objects on the basis of created patterns in program mentioned above and static patterns prepared separately.

The patterns generator was implemented in C++ by using Unreal Engine environment in version 4.0 [6, 7]. The object which has to be detected by the system is loaded as a 3D model. A skeleton and a description of these parameters allow the patterns generator to present the object in many variations. For example, in a gesture detection system, a program will load a hand model, where the following will be modifiable in the operating time: the hand's size, the location of the fingers, the hand's color and the variant size of each finger (Fig. 1). These changes can be made randomly

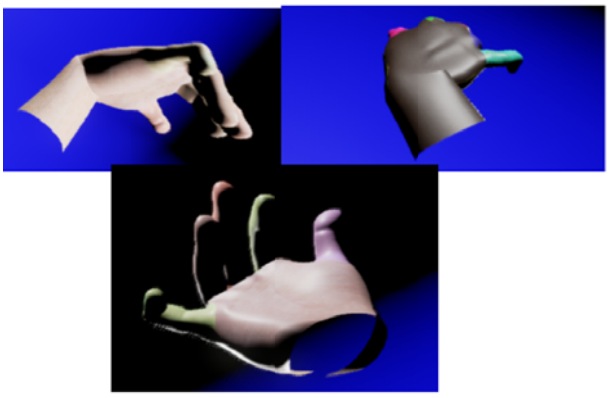

**Fig. 1 Example of dynamically generated learning patterns on the basis of a 3D model.**

or by using previously set of parameters.

The learning application was implemented in C++ language in Visual STUDIO 2015 and QT environment [6, 7, 9]. The results of these works are a creation of two neural networks. The first neural network is designed for detecting an object in the digital image, whereas the second neural network compares the detected image with images of objects in the base data with the intention to get more specific information about the detected object. For example, we can detect a new car by the first neural network but if we want to know the car's brand, we have to compare that car with images of cars of different brands, which are located in the data base. Both the first and the second neural network have one output which gives value in range from -1 to 1 [8]. The object is accepted as a detected or recognized if the output value of a neural network is greater than zero.

### *2.2 Processing Sub-system*

Besides the software component the processing sub-system also has the movable sensor which provides a stream of digital images and an operator who is responsible for the sensor's movements.

The software part of the sub-system contains the data base of objects to recognition and application for image processing which executes detection and a recognition process of the requested object. The thread which provides stream of digital images is an important element of application. The stream can be sent by TCP/IP or UDP protocol. The sending device could be LTE module (in this case if the area of processing is the whole area where we have the communication coverage, for example city) or WI-FI module (in the case of small area) [14].

The movable sensor in this paper is a quad-copter drone. The drone is controlled by the human operator or by an algorithm (which at this moment is in an implementation phase). Except for the parts that are vital for flying (battery, motors, microcontroller for flying control), the drone has a mounted camera and a Raspberry Pi ver. 2.0 microcontroller [13] which provides the full control and drone programmability. The camera provides the stream of digital images [16, 17] which is used for processing [9].

## 3. Algorithms Used in the Learning Sub-system

The most important algorithms used in the process of neural network learning and the algorithms of initial preparation and generating learning data are as follows:

### *3.1 Preparing Learning Data*

All the input data (input images) are subjected to following operations:

- resizing (smaller resolution);
- conversion to shades of grey;
- histogram equalization;
- normalization (Fig. 2).

Changing the size of input image to a smaller size allows speeding up the process of learning (the architecture of neural network is simplified, having less

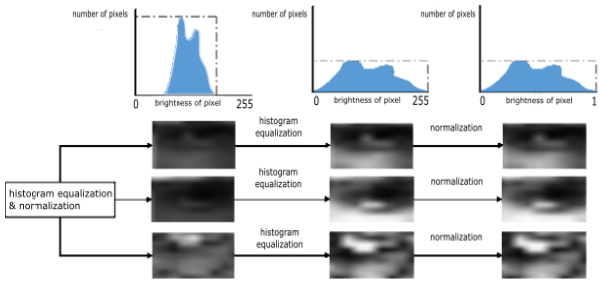


**Fig. 2 Histogram and normalization of input images.**

inputs and connections between layers which results in less floating-point calculations). In the case of face detecting neural network of sufficient image resolution is 20 × 20 pixels [8]. Face recognizing neural network uses image resolution of 40 × 40 pixels, which is sufficient for face comparison. Image conversion to shades of grey and histogram equalization allows the system to act independently from the lighting type and the requested object's color. It means that neural network is concentrated on learning about general features of object and not about its special cases. Normalization is vital in the learning process because it prevents overflows and surfeits of modified weights values [8, 10, 11].

### *3.2 Learning of Detecting Neural Network*

The detecting neural network has the three-layered, feed forward structure with no-linear activation function. The learning process is conducted by using the back propagation algorithm with momentum with adjusting learning factor [10]. Selection of learning patterns is realized by a Bootstrap algorithm which allows to decrease the number of patterns needed for learning process. The initial size of learning patterns is small. The next learning patterns are added periodically during the learning process after some iterations of back propagation algorithm. The pattern is added only when it is wrongly recognized by a neural network because there is no reason for adding patterns correctly recognized to training set (Fig. 3). Thanks to this the neural network learns new features of the object analyzed instead of duplicating what is had already learned. It speeds up the single iteration of algorithm learning, because it eliminates processing patterns that do not contribute to the learning process [1, 8, 10, 11].

### *3.3 Learning Image Comparing Neural Network*

The neural network used in the process of comparing images has a similar architecture to the detecting neural network. The difference is that first and second layer's connections are not type each to each. Connections

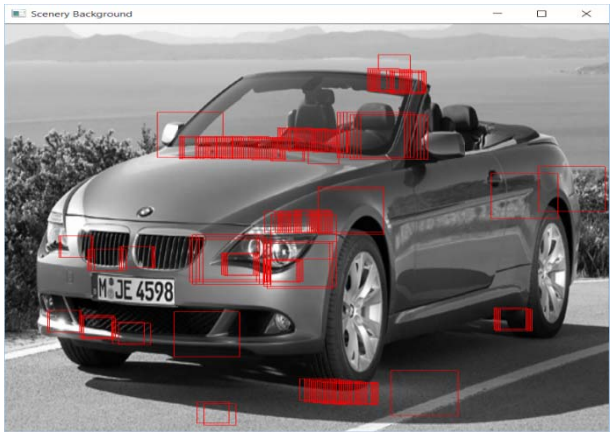


**Fig. 3 Example of adding negative patterns during the time of learning neural network for face detection [8, 15].**

were selected so that one half of neurons will be processing one image and the second half of neurons will be processing the second image. This operation allows to speed up the learning process in contrast of using connections each to each. It decreased amount of executed floating point calculations what means the process speeding-up of image propagation in the neural network [5, 8]. This learning process also uses a Bootstrap algorithm. Additionally, next to the learning process, the probability thread is executed, which provides us information about the effectiveness of comparing images.

### *3.4 Generation and Processing of Static Patterns*

In the initial stage of neural network learning takes place the learning process to identify which elements are constant and unchangeable. In order to increase the generalization capabilities of a neural network, elements of data patterns are subjected to following operations:

- mirror reflection (horizontally);
- rotation (in range of 5 ÷ 10 degrees);
- size changes (an increase or a decrease) in range of 5÷10% of the base size.

Size and rotation degree are randomly selected [8]. In the next step elements of data patterns are converted to form, which is accepted by the input of neural network (Fig. 4). A conversion is conducted by the operations described in the point one of this chapter [8].

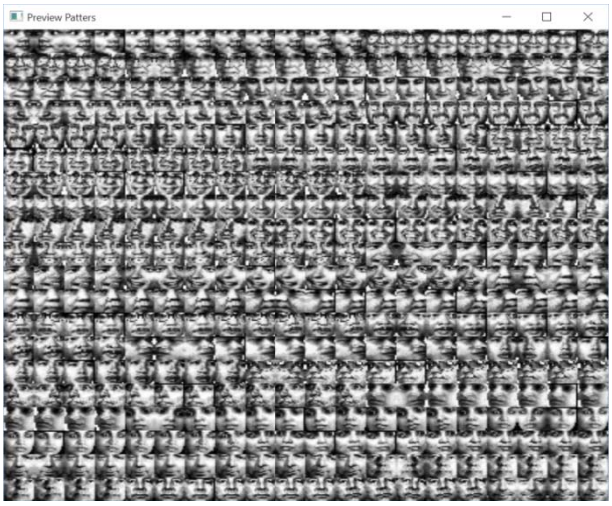


**Fig. 4 Window displayed during the learning process with processed patterns of training set [8, 15].**

### *3.5 Generation and Processing of Dynamically Selected Patterns*

The dynamical patterns data are created during learning time and depend on the current learning progress of neural network. In contrast to static training set, the dynamic patterns are generated based on the 3D model. Thus, we have possibility to modify object's location in relation to the camera, object's texture, background changes, modification method (randomly, by steps), noise insertion [8]. There are practically unlimited possibilities and depending on the parameters definitions of patterns generation process.

## 4. Algorithms Used Processing Sub-system

This chapter describes algorithms used for processing images, detecting and recognizing objects.

### *4.1 Searching in Image Space*

To make it possible for neural network to detect objects in different sizes, every image frame provided by drone has to be subjected to sub-sampling operation. On the basis of original frame we create images of different diminishing size. Each of these created images is processed by the shifting window which resolution could be handled by input of neural network. Obviously before presenting image fragment from shifting windows to the input of neural it has to be normalized and subjected to operations described in chapter 3.1.

### *4.2 Efficient Histogram Equalization*

To allow the system to operate in real time it was necessary to speed up the histogram calculation of every image fragment from shifting windows. In contrast to a standard algorithm, the complete histogram is calculated only once at the beginning. Histograms for next shifting windows are calculated on the basis of the histogram from previous shifting window. For example, for detecting purposes our shifting window has a resolution of $20 \times 20$ pixels. If we want to calculate histogram we have to process 400 elements (pixels) of the analyzed image fragment. However, by using an efficient histogram, for every next shift window (by moving left, right, down per pixel length) we have to process 20 elements of entry column (add these values to histogram) and 20 elements of an outgoing column (subtract these values from histogram). As we can see, an efficient histogram remarkably decreases the amount of elements necessary to process and change search operations on matrix values into simple adding and subtracting operations on the small vectors [1].

## 5. Used Technologies

This chapter describes the hardware technologies in use, which were required in the process of achieving real-time processing detection and object recognition and to accelerate the learning process.

### *5.1 Processing on the Graphic Card (GPU)*

Thanks to shader units graphics card processor is able to process simultaneously a lot of threads operating on the large matrices of data. The basic requirement for accelerating the algorithm with the use of a GPU is the possibility to rewrite the algorithm to his parallel version. The amount of threads which we can execute at the same time is about 10,000 and more and this efficiency is not possible to be achieved at

CPU, where the limit of current time allows us to run the program using eight threads simultaneously.

The back propagation algorithm used in the learning sub-system was modified and adjusted for executing on the GPU processor. All patterns are being processed simultaneously in one epoch of the learning algorithm. It accelerated the learning process by 2 to 3 times in the contrast to algorithm which proceeded at the CPU (central processing unit), where the learning patterns are processed sequentially.

There is a possibility to speed up even 5-10 times relatively to CPU, but in this case there is a necessity to use shared memory of thread blocks.

The technology which allowed to implement the learning algorithm at GPU is C++ AMP [4].

### *5.2 Multi-thread Processing*

Depending on the capabilities of the available CPU, there is a theoretical possibility of *n*-multiplying increase to speed up the program where *n* is the number of processor's cores.

The image processing algorithm was modified by dividing it into 4 threads, which means that the image is now divided on four processing areas. Every thread is responsible for one of the four parts of image. Additionally, for calculating outputs of the neural networks and histogram equalization there were used SSE (Streaming SIMD Extensions) instructions, which allow to process simultaneously up to four math operations.

By using multi-threading (4-threads) and SSE instructions it was possible to achieve an increase from the average time of about 250 to 10-20 milliseconds. That is an improvement of about 25 times.

## 6. Results

This chapter shows results of calculations described in this paper for used algorithms.

### *6.1 Comparison of Speed Learning at GPU with CPU*

Figs. 5 and 6 present amount of iterations done by learning algorithm in 3 hours' time for GPU and CPU.

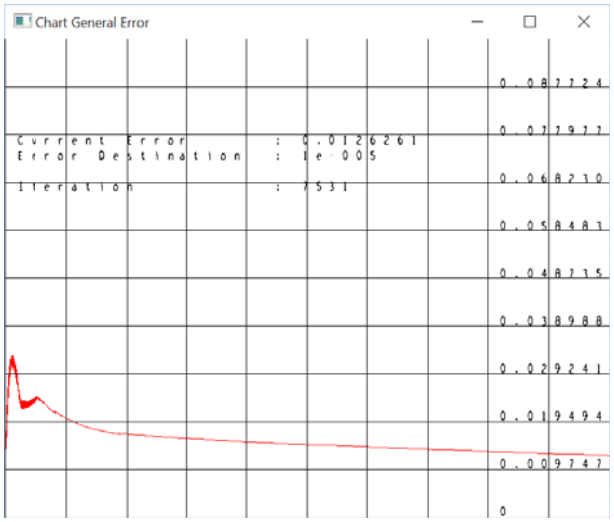


**Fig. 5 General error at CPU for 3 hours of learning.**

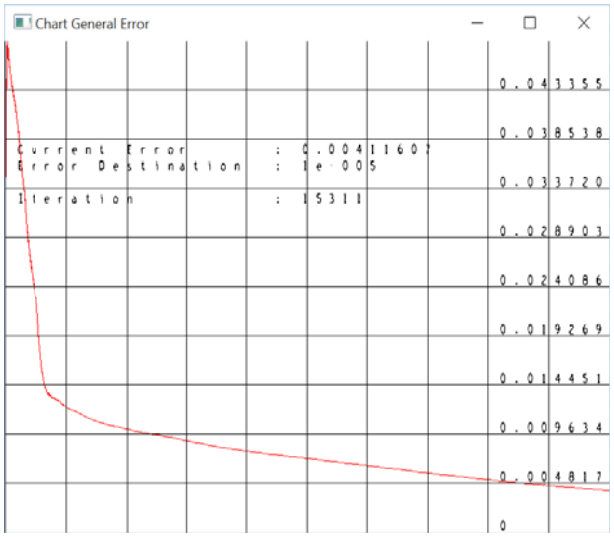


**Fig. 6 General error at GPU for 3 hours of learning.**

Fig. 6 shows that speed of learning at GPU is much greater than in the case of executing this algorithm at CPU. If we calculate ration of amount operations of GPU to the one performed at CPU we notice 15311/7531≈ **2**-times faster execution of algorithm.

### *6.2 Comparison of Multi-threading Processing with Single Thread Processing and with SSE or without*

Table 1 shows results of comparison of multi-thread with single thread processing.

The results of a multi-thread processing with SSE show the multiple speed-up of processing sub-system.

**Table 1 Comparison of multi-thread with single thread processing.**

| Speed of processing of single frame video stream | | |
|---|---|---|
| Thread amounts | 1 without SSE | 4 with SSE |
| Standard histogram | 250 ms | 130 ms |
| Efficient histogram | 100 ms | 16 ms |

If we divide second by time of processing of single frame we get theoretical speed of processing which is 62 fps. The achieved speed-up is sufficient to be used by the sub-system in the real-time processing.

## 7. Conclusions

The results confirm that nowadays not only the knowledge concerning rules of operating efficient algorithms but also the knowledge about hardware platform which will be used for implementation purposes is very important. Familiarity with multi-thread processing and skills in converting seemingly serial algorithms to parallel version is fundamental for creating efficient and fast algorithms which we can use in the real time processing systems.

By using AI, we allow our system to acquire knowledge about features of the analyzed object, which the designer/programmer is not able to predict while designing. In a certain way in the course of the learning process the system is better than its designer. The neural networks allow for the system to get general knowledge about the requested object. It means that the artificial neural network, similarly to biological counterparts, can detect an object in different lighting, presence of interferences, etc. The difference is that digital neurons are never tired and in contrast to biological neurons their quality of processing is always constant.

## References

[1] Efficient Histogram-Based Sliding Window. http://msr-waypoint.com/en-us/people/yichenw/cvpr10_ehsw.pdf (access 05.11.2015).

[2] "Fuzzy Differential Equation for Nonlinear System Modeling with Bernstein Neural Networks." IEEE Access. doi:10.1109/ACCESS.2017.2647920.

[3] Jafari, R., and Yu, W. 2017. "Fuzzy Modeling for Uncertainty Nonlinear Systems with Fuzzy Equations." *Mathematical Problems in Engineering.* Vol. 2017, https://doi .org/10.1155/2017/8594738.

[4] Gregory, K., and Miller, A. *C++ AMP Accelerated Massive Parallelism with Microsoft Visual C++*. O'Reilly Media, Inc, 1005 Graven Stein Highway North Sebastopol, California 95472.

[5] Learning to Compare Image Patches via Convolutional Neural Networks. http://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Zagoruyko_Learning_to_Compare_2015_CVPR_paper.pdf (access 28.05.2017).

[6] Megatutorial–Od zera do gier kodera (ang. Tutorial—From Scratch to Game Coder). http://xion.org.pl/productions/texts/coding/megatutorial/ (access 28.05.2017).

[7] Microsoft Software Developer Network (MSDN), Visual Studio IDE User's Guide. https://msdn.microsoft.com/en-us/library/jj620919(v=vs.120).aspx (access 19.10.2015).

[8] Neural Network-Based Face Detection. http://www.informedia.cs.cmu.edu/documents/rowley-ieee.pdf (access 19.10.2015).

[9] Open Source Computer Vision (OpenCV). http://opencv.org/ (access 28.05.2017).

[10] Osowski, S. 2006. Sieci neuronowe do przetwarzania informacji (ang. Neural networks for data processing). Warsaw University of Technology, Warsaw.

[11] Osowski, S. 1996. Sieci neuronowe w ujęciu algorytmicznym (ang. Algorithms of neural networks). Warsaw University of Technology, Warsaw.

[12] Raspberry PI. https://www.raspberrypi.org/ (access 28.05.2017).

[13] Rozpoznawanie twarzy za pomocą sieci neuronowych (ang. Face recognition by using neural networks). http://www.michalbereta.pl/dydaktyka/KPO/Rozpoznawanie%20twarzy.pdf (access 28.05.2017).

[14] TCP/IP Python Communication. https://wiki.python.org/moin/TcpCommunication (access 28.05.2017).

[15] Tutorials for Modern OpenGL (3.3+). http://www.opengl-tutorial.org (access 28.05.2017).

[16] Wikipedia the Free Encyclopedia, Bitmap. https://en.wikipedia.org/wiki/Bitmap (access 19.10.2015).

[17] Wikipedia the Free Encyclopedia, Digital Image. https://en.wikipedia.org/wiki/Digital_image (access 19.10.2015).