UNIWERSYTET TECHNOLOGICZNO-PRZYRODNICZY
IM. JANA I JĘDRZEJA ŚNIADECKICH W BYDGOSZCZY
ZESZYTY NAUKOWE NR 267
TELEKOMUNIKACJA I ELEKTRONIKA 20 (2017) 5-14

THE USAGE OF NEURAL NETWORKS TO FORECAST FOR CHURN OF TELECOMMUNICATIONS CLIENTS

Przemysław Wojda

Warsaw University of Technology, Faculty of Electronics and Information Technology
Nowowiejska 15/19, 00-665 Warsaw, Poland

Summary. This paper presents an attempt to use an artificial neural network to investigate the churn phenomenon among the customers of a telecommunications operator. An attempt was made to create a data model based on the customer lifetime value (CLV) rather than on activity alone. A multilayered artificial neural network was used for the experiments. The results yielded a 99% successful identification rate for customers in no danger of leaving, while only 57% of those identified as in danger of leaving actually did so and stopped using the company's services.

Keywords: churn, artificial neural network, ANN, CLV, telecommunications

1. INTRODUCTION

The churn phenomenon is defined as discontinuing the use of a company's services. This most commonly means that the customer has chosen another company because it seems more attractive. A customer leaving translates into costs not only in the form of non-generated revenue but also the cost of acquiring a new customer to replace the one who left. Therefore, this phenomenon has no benefit and should be prevented. There are several methods [1, 3, 5] designed to provide an early warning signal for groups of customers in danger of leaving. The decisive factor for their effectiveness is the selection of an appropriate set of data for effective research. The most well-known models focus on statistical data analysis in order to eliminate noise and maximize the usefulness of the variables engaged in classification. However, it is important to know the total value of the CLV, or Customer Lifetime Value [4, 6]. It is not enough to know which customers may leave; it is also important whether keeping them is financially viable in terms of the revenue that they may generate. This study attempts to apply this approach to the company's actual customer base.

2. DATA ANALYSIS

The company is one of the leading company on the Polish market of hosted telecommunication services (SaaS) for small and medium business (SMEs) as well as individuals. It provides IP telephony (voice calls) and value added services like call center,

IVR, calls recorder. The company also provides its own telecommunication platform for other telco providers. It won many awards for its innovative products. The company provides postpaid and prepaid services using two different technological platforms. The services are offered within domains corresponding to independent offers. The data was acquired from the company's data centers and these are call detail records (CDR) and the information about the payments of the customers. To fully understand the figures presented below, the proper definition of CDR should be quoted: *it is a data record produced by a telephone exchange or other telecommunications equipment that documents the details of a telephone call or other telecommunications transaction (e.g., text message) that passes through that facility or device. The record contains various attributes of the call, such as time, duration, completion status, source number, and destination number.*

Based on the number of active users, generated traffic and revenue, the surveyed dataset was narrowed down. As a result, the research covers the period between January 2013 and June 2015 and only includes prepaid services in domain no. 1.

The collected dataset was then analyzed according to customer activity on a monthly basis. Activity in the monthly data is defined as follows:

1. At least one event in terms of either inbound or outbound traffic,
2. At least one financial document registered on the customer's account.

Based on this definition, for each customer who stopped using the company's services during the period under research, the number of months in which he or she remained active was determined. Knowing the customer's last month of activity, the next step was to analyze the characteristics of the traffic generated regarding customers who are still active.

For example, for customers whose last month of service was January 2015, the activity rate for outbound and inbound traffic is as follows (Fig. 1).

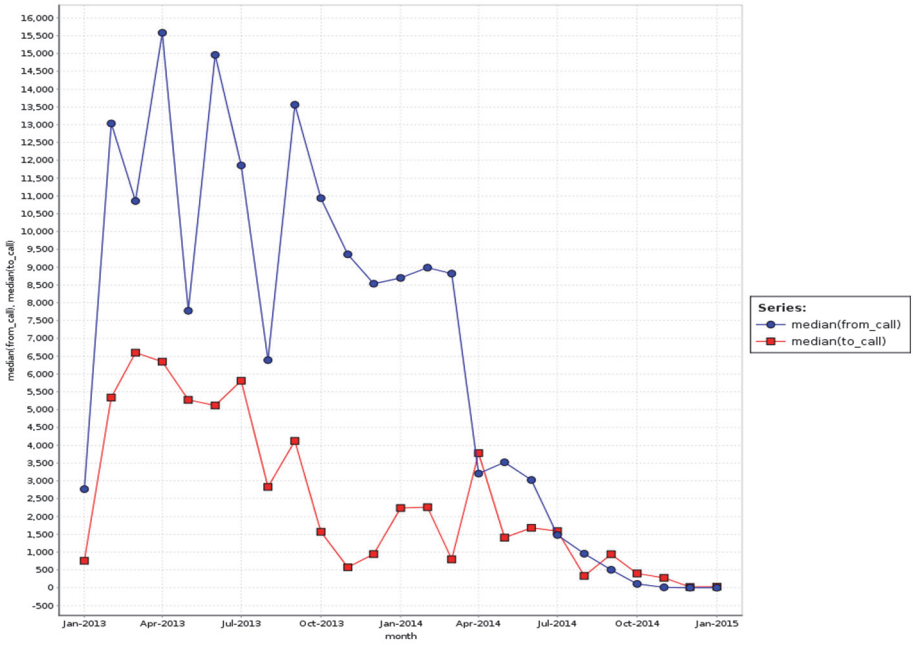

Fig. 1. Median product of duration and number of outgoing and incoming calls as a function of time (Source: own study)

However, for customers who continued using the service (Fig. 2).

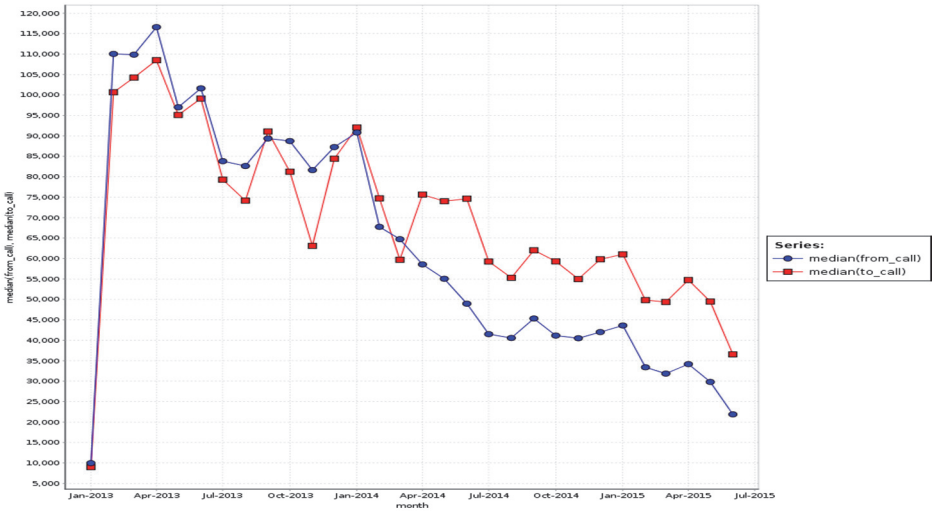

Fig. 2. Median product of duration and number of outgoing and incoming calls as a function of time (Source: own study)

Both graphs show a downward trend (a phenomenon familiar to the company in the case of the service under research), but for active customers, the lines remain far above zero.

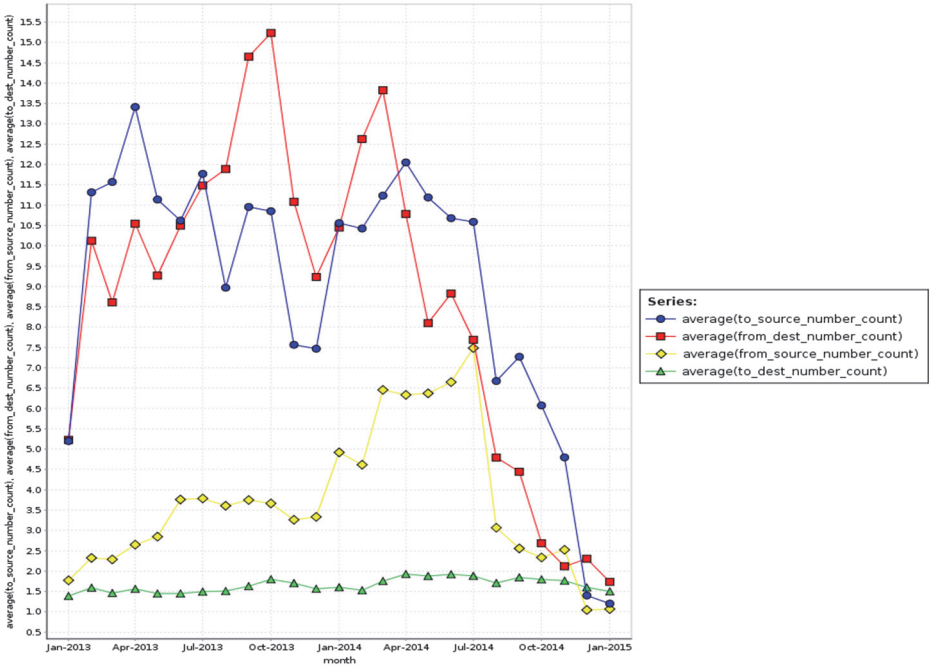

Fig. 3. Average number of different numbers to which connections were made in a given month (Source: own study)

An analogy can be observed by analyzing the variety of connections (the number of different numbers to which the connections were made). For the same period, for customers who left (Fig. 3).

However, for customers who remained active (Fig. 4).

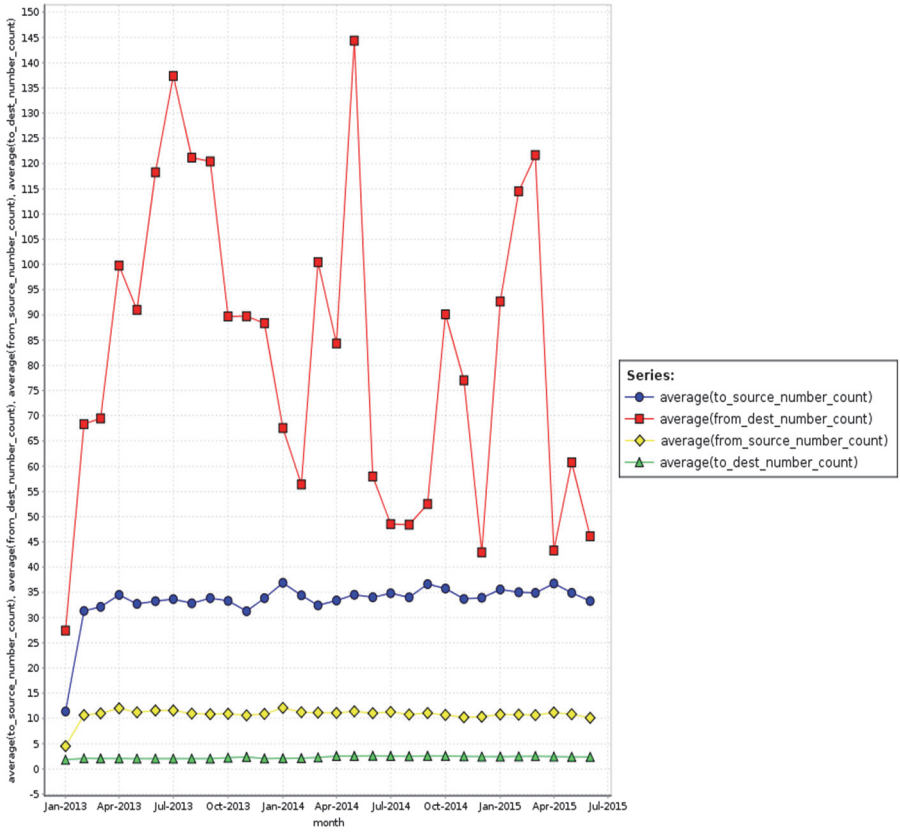

Fig. 4. Average number of different numbers to which connections were made in a given month (Source: own study)

The final conclusions from the data research stage are as follows:

a) the number and duration of connections, both for outbound and inbound traffic approach zero,
b) the number and amount of credit-enhancing as well as debit-related financial documents approach zero,
c) the variety of outbound and inbound connections clearly decreases as the customer's last month of activity approaches,
d) when a customer moves closer towards the end of his or her activity, the traffic is generated by a decreasing number of numbers in the customer's possession.

3. INPUT DATA

Based on the above analysis, the following customer characteristics were established in order to create the neural network model:

a) the number of active months – the number of months from the first month to the last month in which the customer was active,
b) outbound traffic measurement – the product of the number of outbound connections and their duration over a period of one month,
c) inbound traffic measurement – the product of the number of inbound connections and their duration within a month,
d) measurement of traffic within a group of customer numbers – the product of the number of connections between customer numbers and their duration within a month,
e) measurement of the size of the group of customer numbers – for numbers belonging to a customer group, the sum of the numbers of outbound connections and the number of numbers to which inbound connections were made within a month,
f) measurement of the diversity of external numbers – for the numbers belonging to the customer, the sum of the numbers of the outbound connections and the number of numbers from which inbound connections were made within a month,
g) measurement of credit increase – the product of the number of credit-enhancing documents and their amounts within a month,
h) measurement of credit reduction – the product of the number of documents decreasing credit and their amounts within a month,
i) measurement of debit reduction – the product of the number of documents decreasing debit and their amounts within a month.

4. DATA CLEANSING

For the purposes of the cleansing and future categorization of the collected data, the following auxiliary features were introduced:

a) activity – the total number of outbound and inbound connections and number of events increasing credit and decreasing debit in a given month,
b) current activity – sum total since the beginning of a group's activity,
c) current number of the month – the month in which the activity is researched,
d) future activity – activity in the period following the current month under research, within the available data,
e) future number of months – the number of active months in the period following the current month under research, within the available data,
f) variation coefficient [2], defined as the ratio of the standard deviation to the mean of the characteristic considered in terms of the group.

Next, data regarding any groups that met the following criteria were removed from the set:

a) breaks in activity of longer than a month,
b) total known activity less than the mean activity for all groups,
c) total known activity greater than the mean activity for all groups,
d) activity coefficient less than the median for all groups,
e) coefficient of activity variation greater than the mean for all groups.

5. CATEGORIES

For the purpose of assigning the collected data to categories, another feature was defined – group utilization. This attribute is defined as the ratio of the current activity of the group to the total known activity of the group. The value of the feature is within the range of <0; 1>, with a value of 1 reached in the last known month of activity. Based on the available data, the following categories were introduced:

- DEAD – an inactive customer, the utilization measurement equals 1;
- **CHURN** – the customer will become inactive within three months, the utilization measurement is within (0,96;1);
- ALIVE – an active customer.

The key category is CHURN – the correct classification of a customer to this category indicates that the customer is in danger of leaving within three months.

6. EXPERIMENTS

The study was based on a feedforward artificial neural network, multilayer perceptron because of its proven effectiveness in problem classification. Experiments were conducted with varying numbers of neurons and different activation functions in the hidden layer. The influence of constant bias and more than one hidden layer on the results was studied. Different methods of network training were also applied, with particular emphasis on different variants of the reverse resonance algorithm (Manhattan, Resilient).

The most important element of the assessment of the usefulness of the network was the error obtained on the set testing (not used for training). It should be noted that almost 100% matching the network to the teaching set will generally be associated with the phenomenon of over-matching and the error obtained on the test set will be disproportionately large. In order to avoid this phenomenon cross-validation was used. This extends the network training process but the obtained model is less susceptible to excessive fit. In other words, cross-validation was used to evaluate the results of the network, while the goal was to minimize the mean squared error during training. Finally, the best results (7.21% error in the test set, 84.79% of the examples correctly classified) were obtained with a network with the following parameters (Table 1).

Table 1. Network parameters

Layer	**Number of neurons**	**Activation function**	**Bias**
1	26	Linear	No
2	11	Logarythmic	Yes
3	3	Softmax	Yes

Table 2 shows the results for the test set used (the number of examples depending on the category allocated).

Table 2. Network topology

Actual category	Allocated category	Share in test set (%)
ALIVE	ALIVE	13,807531
ALIVE	CHURN	0,13947
ALIVE	DEAD	0
CHURN	**ALIVE**	**0,13947**
CHURN	**CHURN**	**11,018131**
CHURN	**DEAD**	**8,089261**
DEAD	ALIVE	0
DEAD	CHURN	6,834031
DEAD	DEAD	59,972106

The results are shown in the Fig. 5.

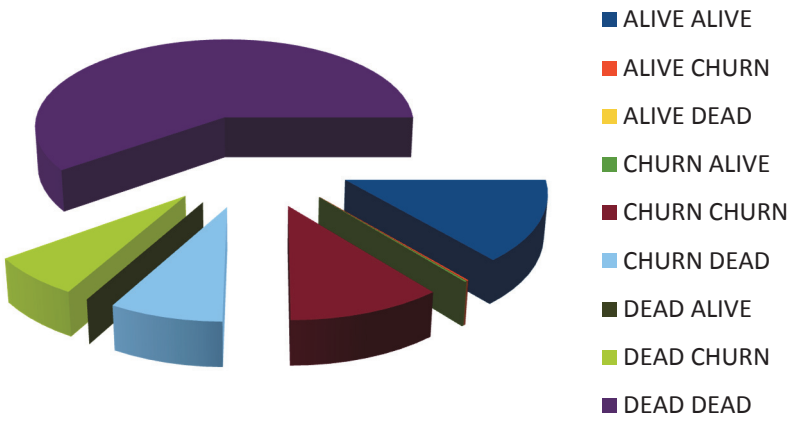

Fig. 5. Summary of the obtained results (Source: own study)

The observed error is mainly due to misclassification:

a) CHURN as DEAD (8,09% of cases),
b) DEAD as CHURN (6,83% of cases).

A positive aspect is the correct recognition of the ALIVE category.

Table 3. The percentage of correctly classified examples within a given class

Class	Correct classification (%)
ALIVE	99
CHURN	57,24
DEAD	89,77

7. FINAL CONCLUSIONS

The results are debatable. On the one hand, almost 99% of the customers who did not plan to stop using the service in the next three months were identified. On the other hand, among the group identified as customers in danger of leaving only 57% did in fact stop using the service. The key factor in obtaining the results was to define the model used for the study. According to the author, the model created is applicable to a rough screening of cases that do not raise any suspicion. More precise forecasts require the creation of separate, event-specific models focused on the particular phenomena that influence a customer's decision to move to another service provider. The use of such a combined approach (a rough identification of the risk group followed by an examination of specific symptoms) would result in a high precision churn forecasting tool.

The churn phenomenon has many aspects and the presented study was just an attempt to investigate the problem from a selected perspective (customer activity). However, the author is convinced that it is possible to use neural networks for detecting the phenomenon of churn. The research carried out shows that the most important is the way of presenting the problem to be solved. Universal classification possibilities (or regression) offered by the neural network are not a limitation here and their use is in the key issue of implementation.

This study can be used as a first step towards creation of a tool that allows the practical use of neural networks for recognition complex dependencies in the data about the company's clients. According to the author, this requires conducting extensive analyzes aimed at determining specific phenomena and patterns behaviors that are in the sphere of interest. This requires knowledge of specifics of the products and the way they are used.

BIBLIOGRAPHY

[1] Ahmed A., Linen D.M., 2017. A review and analysis of churn prediction methods for customer retention in telecom industries.
[2] Abdi H., 2010. Coefficient of Variation.
[3] Ljungehed J., 2017. Predicting Customer Churn Using Recurrent Neural Networks. School of Computer Science and Communication, Stockholm, Sverige.
[4] Hosseni M.B., Tarokh M.J., 2011. Customer Segmentation Using CLV Elements. Information Technology Group, Industrial Engineering Department, K.N. Toosi University of Technology, Tehran, Iran.

[5] Vafeiadis T., Diamantaras K.I., Sarigiannidis G., Chatzisavvas K.Ch., 2015. A comparison of machine learning techniques for customer churn prediction.
[6] Zhang Y., Qi J., Shu H., Li Y., 2006. Predicting Churn Probability of Fixed-line Subscriber with Limited Information: A Data Mining Paradigm for Enterprise Computing. [In:] Tjoa A.M., Xu L., Chaudhry S.S. (eds.), Research and Practical Issues of Enterprise Information Systems. IFIP International Federation for Information Processing, vol. 205, Springer, Boston, MA.

WYKORZYSTANIE SZTUCZNYCH SIECI NEURONOWYCH DO PROGNOZOWANIA ZJAWISKA CHURN WŚRÓD KLIENTÓW USŁUG TELEKOMUNIKACYJNYCH

Streszczenie

W pracy przedstawiono próbę wykorzystania sztucznej sieci neuronowej do badania zjawiska churn wśród klientów operatora telekomunikacyjnego. Podjęto próbę stworzenia modelu danych opartego o całkowitą wartość klienta (CLV), a nie tylko jego aktywność. Do przeprowadzenia eksperymentów wykorzystana została wielowarstwowa sztuczna sieć neuronowa. Uzyskano 99% skuteczność identyfikowania klientów nie zagrożonych odejściem, natomiast tylko 57% klientów wskazanych jako zagrożonych odejściem w rzeczywistości zaprzestało korzystania z usług firmy.

Słowa kluczowe: churn, sztuczne sieci neuronowe, ANN, CLV, telekomunikacja

UNIWERSYTET TECHNOLOGICZNO-PRZYRODNICZY
IM. JANA I JĘDRZEJA ŚNIADECKICH W BYDGOSZCZY
ZESZYTY NAUKOWE NR 267
TELEKOMUNIKACJA I ELEKTRONIKA 20 (2017) 15-24

OBJECT TRACKING METHODS COMPARISON

Daniel Bujnowski, Agata Giełczyk

UTP University of Science and Technology,
Faculty of Telecommunications, Computer Science and Electrical Engineering,
Al. prof. S. Kaliskiego 7, 85-796 Bydgoszcz, Poland
daniel.bujnowski@utp.edu.pl, agata.gielczyk@utp.edu.pl

Summary. Object tracking has been improved recently and now it seems to be one of the most challenging task in a computer vision area. In this article there are presented five top state-of-art algorithms. There were tested and the comparison of their results was performed and presented in plots and tables. A precision and an accuracy were evaluated, while some intruding factors, like rotations or blurring, were observed.

Keywords: object tracking, image processing, artificial intelligence

1. INTRODUCTION

Visual tracking is one of the most crucial problems in the computer vision. It can be a very time consuming process and due to this fact, has to be developed. Visual tracking may be applied in various systems like: in a traffic management system as stated by Coifman *et al*. in [4], in medical diagnostics as described by Yang *et al*. in [19] or in some security issues like tracking an object by an unmanned aerial vehicle proposed by Pestana *et al*. in [13]. The applications are crucial for security and health issues, thus is a critical problem in the computer vision. Tracking algorithms have to be robust to different factors that may appear. According to Yang *et al*. [18] the following things may intrude the tracking process:

1) motion blur,
2) noise in an image,
3) representing the real 3D world in the 2D image,
4) covering or partially covering the object,
5) illumination changes,
6) scale change,
7) deformations,
8) real-time processing requirements.

Due to the variety of the object tracking methods, the accuracy and the precision researches were performed to choose the most appropriate one. Moreover, above mentioned intruding factors were analysed.

The paper is organized as follows: in Section 2 popular tracking algorithms are presented. Section 3 contains the description of researches done with the results presented in plots and tables. Conclusions are provided afterwards.

2. OBJECT TRACKING ALGORITHMS

2.1. Main steps of image analysis

The most basic steps of image processing are similar for many applications. They are also implemented in object tracking and may be presented as following [12, 14]:

1. Image acquisition – image can be delivered to the system either from the camera or from the file.
2. Pre-processing – there are simple morphological operations that may improve the image or reduce noises like: filters, erosion, dilation, wave transformations. This step may include also image segmentation when each image is divided into classes, the most basic situation is using two classes: a background and a foreground. Segmentation may be performed in the simplest way by thresholding.
3. Features extraction – based on the pre-processed image it is possible to find some key points and find distinguishing features.
4. Recognition and matching – using the features it is possible to identify the object visible in the image.

2.2. Kernelized Correlation Fiter and Dual Correlation Filter

Kernelized Correlation Filter (KCF) and Dual Correlation Filter (DCF) were proposed by Henriques et al. in 2012 and developed in 2015 [9, 10]. The clue of them is using a special kind of the Toeplitz matrix, the circulant matrix (presented in Eq. 1.) to improve the detection in detection-based tracking. After giving the target parameters (i.e., position) in the first frame, the tracked region is fixed around the target. The region is 2.5 times bigger than the object but provides not only positive samples. Then, the features are extracted from the whole region. Each channel is weighted by a cosine window, which smooths the borders. A circulant matrix is essential for finding all possible either horizontal or vertical shifts of the target, which are presented in Fig. 1.

$$C(x) = \begin{bmatrix} x_1 & x_2 & x_3 & \cdots & x_n \\ x_n & x_1 & x_2 & \cdots & x_{n-1} \\ x_{n-1} & x_n & x_1 & \cdots & x_{n-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_2 & x_3 & x_4 & \cdots & x_1 \end{bmatrix} \tag{1}$$

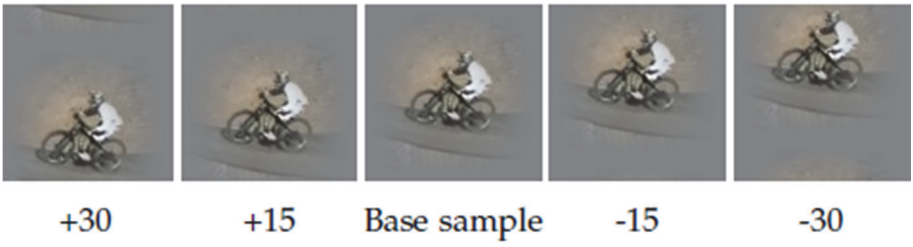

Fig. 1. Possible vertical shifts with the cosine window enhancement (Source: own study)

Thanks to their robustness and effectiveness KCF and DCF are widely implemented. MATLAB codes were provided in papers in order to enable faster progress in object tracking area. Thus, there are plenty of algorithms extensions available [15, 21].

2.3. Compressive Tracking

Due to some factors like position variation, illumination variation or motion blur, tracking algorithms have been constantly developed. The Compressive Tracking (CT) algorithm was proposed by Zhang *et al*. in 2012 [20]. It is supposed to be robust to all above mentioned factors. The main idea of the algorithm is presented in Fig. 2.

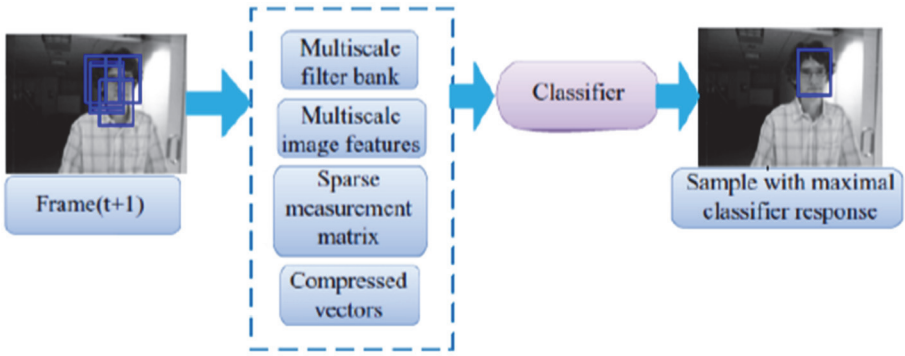

Fig. 2. The main idea of CT algorithm [20]

The problem of tracking is presented as a detection task in the CT algorithm. In the first frame the object detection is performed. Then positive samples are searched in the nearest neighbourhood of the detected object. The locations far away from the object are assumed as negative samples. Both positive and negative samples are used to update the classifier. In order to ensure the robustness to the scale variation, each sample is convolved with the multiscale rectangle filters and to improve the effectiveness the Haar-like features [16] are compressively sensed with a sparse measurement matrix.

2.4. Structured Output Tracking with Kernels

Structured Output Tracking with Kernels (Struck) algorithm was proposed by Hare et al. in 2011 and developed in 2014 and 2016 [6-8]. This algorithm differs from the

other state-of-art methods in learning approach. Commonly, a set of samples is generated and then its type is graded (positive or negative). The Struck approach does not use these steps, but uses only the output, which is illustrated in the Fig. 3A.

The algorithm uses six types of Haar-like features, which are presented in Fig. 3B. Numbers visible in the image are not normalised weights used for calculating the features. Then, features are normalised in the range $[-1,1]$ and stored in a features vector. The length of the vector is 192.

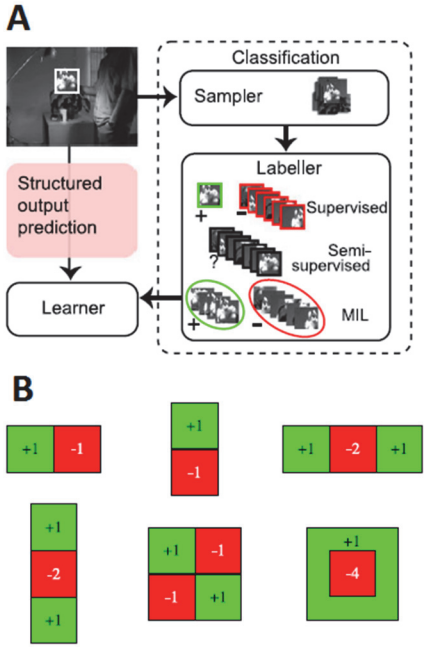

Fig. 3. A) The difference in approach between Struct algorithm and other state-of-art methods [14], B) Types of Haar-like features used by Struct [6]

2.5. Multiple Instance Learning

Multiple Instance Learning (MIL) is a variation of supervised learning. The algorithm was proposed by Dietterich in 1997 [5]. The main idea of this algorithm is using sets, often called bags, during training. Boolean labels are assigned to the whole bag instead of a single instance. When the bag has at least one positive instance, it is positive (value 1). Otherwise, while the bag has not any positive instance, it is negative (value 0). The keychain example, which is illustrated in Fig. 4, may be useful for the MIL algorithm presentation. Each person has their own chain of keys. However, not everyone is able to enter each room. The problem to solve is finding the proper key that enables an entrance to the room. To solve this problem it is needed to find the key that is included in all positive chains.

Training data contains examples $\{x_1, x_2, \dots, x_n\}$ and labels $\{y_1, y_2, \dots, y_n\}$, where $x_i \in X$ and $y_i \in Y$. X is the set of examples, while $Y = \{0,1\}$. The aim is to learn the classifier in order to predict the proper labels for new instances outside the collection X,

using the function $h(X): X \rightarrow Y$ [2]. Currently, datasets are prepared carefully to provide the best training examples. The object is cropped to its borders in order to distinguish the foreground from the background. The object has to be analysed also in a different scale or from a different point of view.

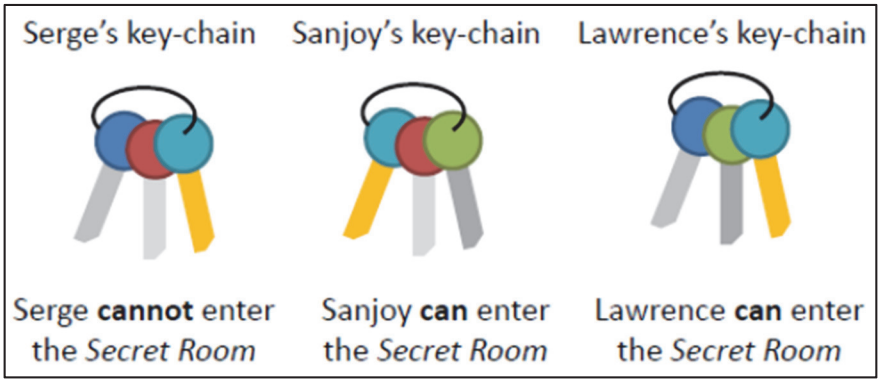

Fig. 4. The main idea of MIL presented in the keychain example [1]

The MIL algorithm is widely used thanks to its effectiveness. Recently, it was implemented for instance in medicine for mammogram classification [22], for genetic RNA analysis [3] or for text classification [11].

3. RESULTS

3.1. Evaluated parameters

The abovementioned algorithms were implemented in MATLAB and tested. The dataset used during the research was the Visual Tracker Benchmark available online [17]. It contains 100 examples of image sequences. The text file, which contains the position of the target (along x and y axis) and the size of the target (width and height), is also provided. Each sequence has some intruding factors like: illumination variation, motion blur or scale variation and others.

Sequences were evaluated due to three parameters. The first one was the distance between the detected position and real position expressed in pixels. An ideal case is having the distance equals to 0. The second one was the Pascal parameter, which is expressed in Eq. 2. Each frame was marked as positive when $a_0 > 0{,}5$. Otherwise, it was negative. The most desirable situation is having the parameter close to 1.

$$a_0 = \frac{area\,(detected \cap real)}{area\,(detected \cup real)} \tag{2}$$

The last but not least parameter was precision, which was evaluated for the whole sequence. It is presented in Eq. 3, where n – a total number of frames in the sequence, i – number of frames with the distance between the real and the detected position is in a range $[0, r]$ while $r \in [1{,}50]$.

$$treshold = \frac{i_r}{n} \tag{3}$$

An example of the analysed sequence is presented in Figure 5. In the sequence called Surfer there are five factors which are making the tracking process harder: scale variation, fast motion, low image resolution, target rotation inside and outside the image.

Fig. 5. Frames 0001 (A), 0075 (B), 0150 (C), 0225 (D), 0300 (E) and 0375 (F) of the sequence Surfer [17]

3.2. Obtained results

During the experiment all before mentioned algorithms were tested against each other. The experiment was performed on PC (CPU: Intel Core i7-6700hq 2.6GHz, RAM: 16GB, GPU: GeForce GTX 960M). Processing evaluations relied only on CPU. The results are presented in Table 1. Moreover, for each sequence, the set of three plots were created (presenting distance, Pascal and precision parameters).

Table 1. Obtained results

	Distance [px]	**Pascal [%]**	**Precision [%]**
Struct	25,33	65	67
KCF	**12,71**	**66**	**69**
DCF	27,00	63	64
CT	50,58	31	29
MIL	48,73	40	39

Three plots presented in Fig. 6 refer to the sequence tracking the helmet of the soccer player. Some frames taken from this sequence are also presented in Fig. 6. Each algorithm works almost in the same way till the moment of crash between the tracked helmet and the other similar one. Then, some algorithms started to track the helmet of the opposite player. Precision is also very similar, most of algorithms get over 60%.

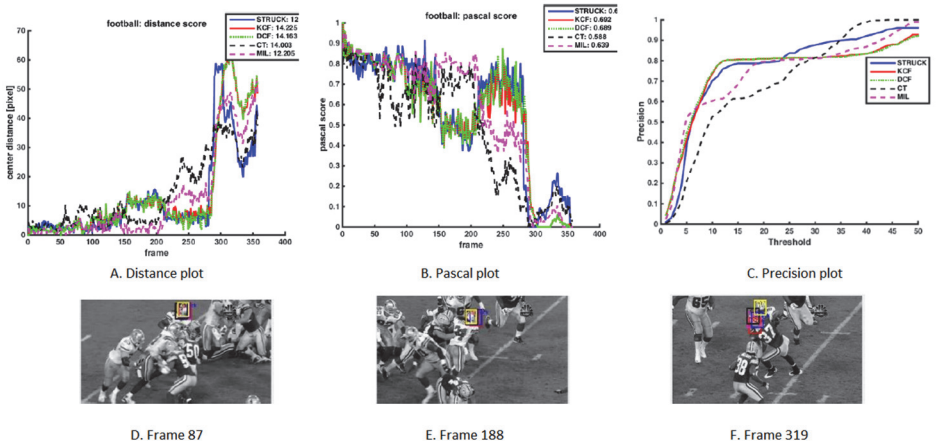

Fig. 6. A) Plot of distance, B) Plot of Pascal parameter, C) Plot of precision, D), E), F) sample frames from the sequence *soccer* (Source: own study)

Plots and frames presented in Fig. 7 refer to the most problematic sequence. Some frames taken from this sequence are presented in Fig 7. In the sequence there were some intruding factors like: illumination variation, object's size changing, specific light (coming from the background of the frames) and angle variations. Due to this factors, none of tested algorithms was able to track object successfully. The average parameters were 28% of Pascal and 13% of precision. Unfortunately, most algorithms were not able to detect the object's size variation. That is why the Pascal parameter has that low value.

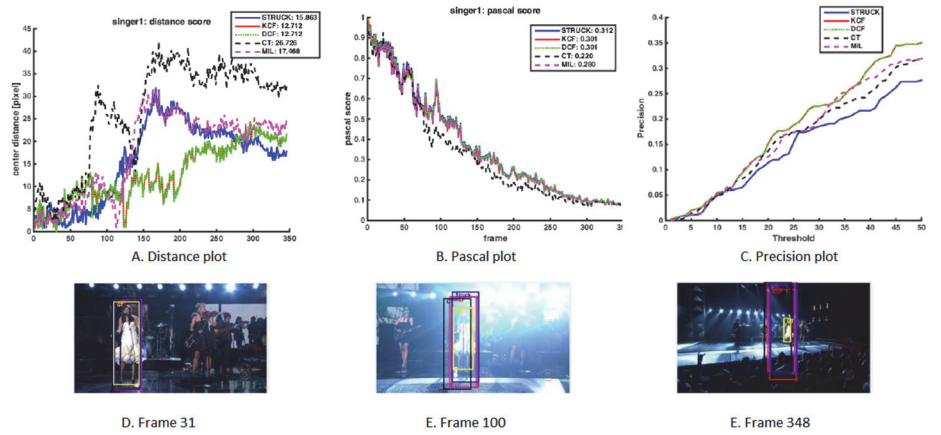

Fig. 7. A) Plot of distance, B) Plot of Pascal parameter, C) Plot of precision, D), E), F) sample frames from the sequence *singer* (Source: own study)

The last but not least group of three plots presents sequence with the moving car. In the frame 160 the car was hidden behind a tree. It is visible that algorithms MIL and Struct missed the tracked object. However, the Struct algorithm was able to detect the object again faster.

The most desirable algorithm should give the minimal distance and in the same time: maximal Pascal parameter and precision. As it is visible in Table 1, the best results were obtained using the KCF algorithm.

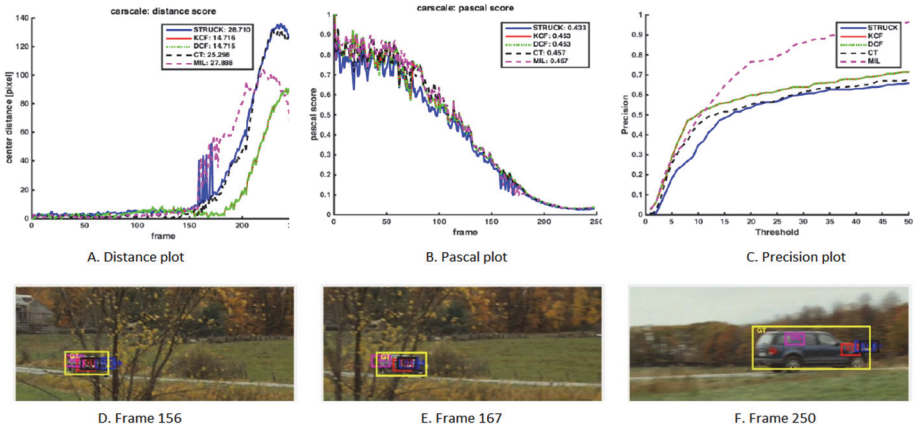

Fig. 8. A) Plot of distance, B) Plot of Pascal parameter, C) Plot of precision, D), E), F) sample frames from the sequence *carscale* (Source: own study)

4. CONCLUSIONS

The performed experiment shown algorithm KCF as the most accurate form all tested algorithms. It had the higher precision rate = 69%, the highest Pascal parameter = = 66% and the smallest distance parameter = 12.71. Moreover, the standard deviation for distance was only 6 pixels, while the others gave standard deviation greater than 30 pixels. Apart of accuracy, it is noticeable that MIL algorithm was significantly slower than rest of tested algorithms. It used over one second more for processing of a single frame.

Nevertheless, the 69% level of precision obtained using the KCF algorithm is still not enough for some crucial applications – security, borders controlling or the military purpose. That is why the object tracking methods should be developed. It seems to be also very promising to combine the object tracking methods with the colour tracking.

BIBLIOGRAPHY

[1] Babenko B., 2008. Multiple instance learning: algorithms and applications. View Artic. PubMedNCBI Google Sch.

[2] Babenko B., Yang M.-H., Belongie S., 2009. Visual tracking with online multiple instance learning. [In:] Computer Vision and Pattern Recognition, CVPR 2009. IEEE Conference on. IEEE, 983-990.

[3] Bandyopadhyay S., Ghosh D., Mitra R., Zhao Z., 2015. MBSTAR: multiple instance learning for predicting specific functional binding sites in microRNA targets. Sci. Rep. 5, 8004.

[4] Coifman B., Beymer D., McLauchlan P., Malik J., 1998. A real-time computer vision system for vehicle tracking and traffic surveillance. Transp. Res. Part C Emerg. Technol. 6, 271-288 .
[5] Dietterich T.G., Lathrop R.H., Lozano-Pérez T., 1997. Solving the multiple instance problem with axis-parallel rectangles. Artif. Intell. 89, 31-71.
[6] Hare S., Golodetz S., Saffari A., Vineet V., Cheng M.-M., Hicks S.L., Torr P.H.S., 2016. Struck: Structured Output Tracking with Kernels. IEEE Trans. Pattern Anal. Mach. Intell. 38, 2096-2109.
[7] Hare S., Saffari A., Golodetz S., 2014. Struck: Structured Output Tracking with Kernels. IEEE Trans. PATTERN Anal. Mach. Intell.
[8] Hare S., Saffari A., Torr P.H.S., 2011. Struck: Structured output tracking with kernels. Proc IEEE Int Conf Comput Vis. 263-270.
[9] Henriques J.F., Caseiro R., Martins P., Batista J., 2012. Exploiting the circulant structure of tracking-by-detection with kernels. [In:] European conference on computer vision. Springer, 702-715.
[10] Henriques J.F., Caseiro R., Martins P., Batista J., 2015. High-speed tracking with kernelized correlation filters. IEEE Trans. Pattern Anal. Mach. Intell. 37, 583-596.
[11] Kotzias D., Denil M., de Freitas N., Smyth P., 2015. From Group to Individual Labels Using Deep Features. Presented at the .
[12] Nigam A., Tiwari K., Gupta P., 2016. Multiple texture information fusion for finger-knuckle-print authentication system. Neurocomputing. 188, 190-205.
[13] Pestana J., Sanchez-Lopez J.L., Saripalli S., Campoy P., 2014. Computer vision based general object following for gps-denied multirotor unmanned vehicles. [In:] American Control Conference (ACC), IEEE, 1886-1891.
[14] Sherawat H., Dalal S., 2016. Palmprint Recognition System Using 2-D Gabor and SVM as Classifier. IJITR. 4, 3007-3010.
[15] Solis Montero A., Lang J., Laganiere R., 2015. Scalable kernel correlation filter with sparse feature integration. [In:] Proceedings of the IEEE International Conference on Computer Vision Workshops, 24-31.
[16] Viola P., Jones M., 2001. Rapid object detection using a boosted cascade of simple features. In: Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on.. IEEE, I-I.
[17] Visual Tracker Benchmark, http://cvlab.hanyang.ac.kr/tracker_benchmark/datasets.html.
[18] Yang H., Shao L., Zheng F., Wang L., Song Z., 2011. Recent zadvances and trends in visual tracking: A review. Neurocomputing. 74, 3823-3831.
[19] Yang L., Georgescu B., Zheng Y., Wang Y., Meer P., Comaniciu D., 2011. Prediction Based Collaborative Trackers (PCT): A Robust and Accurate Approach Toward 3D Medical Object Tracking. IEEE Trans. Med. Imaging. 30, 1921-1932.
[20] Zhang K., Zhang L., Yang M.-H., 2012. Real-time compressive tracking. [In:] European Conference on Computer Vision, Springer, 864-877.
[21] Zhu G., Wang J., Wu Y., Lu H., 2015. Collaborative Correlation Tracking. [In:] BMVC, 184-1.
[22] Zhu W., Lou Q., Vang Y.S., Xie X., 2016. Deep Multi-instance Networks with Sparse Label Assignment for Whole Mammogram Classification. ArXiv Prepr. ArXiv161205968.

PORÓWNANIE METOD ŚLEDZENIA OBIEKTÓW

Streszczenie

Śledzenie obiektów jest coraz bardziej popularne i może być uznane za jedno z najbardziej wymagających zadań w obszarze wizji komputerowej. W pracy zaprezentowano pięć najbardziej wydajnych i najlepiej znanych algorytmów. Zostały one zaimplementowane, przetestowane i porównane. Wyniki tego porównania przedstawiono za pomocą wykresów oraz tabel. Podczas testów oceniane były precyzja i dokładność śledzenia. Obserwowano również wpływ czynników zakłócających na jakość śledzenia.

Słowa kluczowe: śledzenie obiektu, przetwarzanie obrazu, sztuczna inteligencja

UNIWERSYTET TECHNOLOGICZNO-PRZYRODNICZY
IM. JANA I JĘDRZEJA ŚNIADECKICH W BYDGOSZCZY
ZESZYTY NAUKOWE NR 267
TELEKOMUNIKACJA I ELEKTRONIKA 20 (2017) 25-33

SYSTEM APPROACH TO BUILDING FUNCTIONAL PROGRAMS

Vasyl Zaiats[1], Jacek Majewski[1], Beata Marciniak[1], Marii Zaiats[2]

[1]UTP University of Science and Technology, Faculty Telecommunication, Computer Science and Electrical Engineering, Al. prof. S. Kaliskiego 7, 85-785 Bydgoszcz, Poland,
e-mail: zvm01@rambler.ru, vasyl.zaiats@utp.edu.pl ,
[2]Lviv National Polytechnic University, ISM Department, st. S. Bandera 12, 79013 Lviv, Ukraine,
e-mail: zayats.mariya@gmail.com

Summary. The Basic approaches for constructions of the functional programs are considered in the article. The main methods of optimization of the new functions and of the functional are determined. The authors proposed a system approach to building and optimizing the functionality of applications that can be used in the creation of the system of recognizing objects and phenomena. The essence of the approach in a gradual improvement software through software optimization. The developed software is always open and its can be modified, improved or refilled. Each of methods is illustrated by the examples of realization in the environment of Lisp**.**

Keywords: functional program, declarative language, functional defined, Lisp environment, accumulation parameters.

1. INTRODUCTION

We consider approaches to the construction and optimization of newly functional programs to improve the efficiency and reliability of their work. The authors proposed a system approach to creating software for the functional language. The essence of the approach is the constant improvement of the developed software product, its renewal and expansion of functional capabilities with insignificant algorithmic complications, and in some cases be simplification.

The feasibility of using this approach is illustrated by specific application developments are implemented in an environment of standard Lisp [10], which refers to declarative programming languages.

Construction of modern programming languages today is far from perfect. Each of the known languages have their advantages and disadvantages. To determine the usefulness of a language should take into account such considerations:

a) Clarity, simplicity and consistency of language concepts. Obviously, we must to avoid subtle and tricky language restrictions. These restrictions must not be too ambiguous. Semantic clarity of language – is what determines its value.
b) Clarity structure of the program. This requirement must provide syntactic clarity programs written in that language. It should be such as to design other than

semantics, and syntactic records which differed. It is essential that the structure of the program reflects the structure of the algorithm that allowed the development programs the principles of structured programming, software design hierarchy - top to bottom. In this approach, the structure of the program is readily available for the diagnosis, modification and optimization.

c) Naturalness use. This should provide most successful in solving the problem of data structure, to make operations, control structures and easily understandable syntax. This greatly simplifies the creation of software in a given field of knowledge or technical applications.
d) Ease of expansion. Software that are created in that language, can be seen as an extension language. In fact, most programming languages, provides the programmer mechanisms for creating routines. However, the properties of the language itself can facilitate or complicate their use. This ease of expansion the most pronounced in programming languages that have identical presentation of data and applications.
e) External software. This is one aspect that affects the efficiency of the use of language. If we have powerful testing tools, editing, storage, software modifications, then can be made weak language convenient to use, than it without a strong technical support.
f) He effectiveness of creating, testing, transmission, implementation, modification and practical using application programs.

2. ADVANTAGE OF THE FUNCTIONAL LANGUAGES TO RELATION PROCEDURAL

One of the fundamental properties of programming languages, which enables clear calculation described in that language – simple semantics. The great advantage of functional languages like logical that there are some basic concepts, each of which has a simple semantics. In particular, the semantics of functional languages understood in terms of the values that are expressions, but not in terms of action sequences and their use. But from a practical point of view it would be fairer to conclude that a strictly functional language is very elementary and some of its expansion to significantly increased efficiency and clarity of certain classes of deductions. Obviously, it is necessary to distinguish between purely syntactic extensions the semantics requires caution, because it is difficult to understand (clear) already debugged functional programs.

Today, developed hundreds of different programming languages. Even in 1969. Jean E. Sammet [6, 10] gives a list of 120 languages that are quite widely used. This amazing number of programming languages contradicts that most programmers in your practice uses several programming languages, and many of them – one or two programming languages. There is the question for what come to the development of different languages at the unlikely possibility of their implementation. However, if not limited to superficial acquaintance with the language, and have a deep understanding of the concepts underlying the design of programs in that language, then no doubt you can verify the feasibility of the development of various programming languages based on the following considerations:

a) Due to different programming languages study improved understanding of a particular language, its concepts and basic methods and techniques that it uses.

A typical example is the recursion. With proper use it can be elegant, effective program and its application to a simple algorithm could lead to an astronomical increase in time costs. On the other hand, lack of use of recursion in languages such as Fortran, Cobol and understanding of the basic principles and methods of implementation of recursion can clarify the limitations of language, which at first glance is false,

b) The value of programming languages expanding stock of useful programming structures and promotes thinking. Working with data structures of one language, and produce appropriate structure of thinking. By studying other languages and design methods to implement, expand programming thesaurus.
c) Knowledge of several languages allows reasonably choose a programming language for solving a particular problem,
d) The development of a new programming language, like natural language of human communication is always easier if there are several well-known languages,
e) Knowledge of principles of construction of various programming languages facilitates the development of a new programming language.

Construction of modern programming languages today is far from perfect. Each of the known languages have their advantages and disadvantages.

Most modern programming languages are universal, since they give to record any algorithm that language, if not impose restrictions on runtime and capacity of memory, algorithmic complexity, etc. If anyone will offer a new programming language, it is likely to be universal if ignored limits on memory or time. Comparing different programming languages should not proceed with the proportion of what they can do and qualitative differences that define elegance (briefness and clarity), lightness (transparency) and efficiency (speed and hardware) programming them. This comparison should be done in the context of specific applications.

Traditional (algorithmic) programming language versus declarative (descriptive) is quite large and bulky, because do not allow:

a) maximize the capabilities of modern computer technology to ensure the effectiveness of software;
b) clearly visualize algorithms and programs to provide easy inspection and modification of the latter.

Declarative programming languages, which include strict functional programming languages such as S-Lisp [10], R-Lisp [7], Reduce [2], Common Lisp and Auto-Lisp [1] is quite simple and only provide it high enough the severity of programs compared with traditional languages. Several functional programs can effectively run on modern computers, however not as efficient as relevant programs with the assignment operator. This is due to the structure of the architecture of modern computers. In addition, the choice of a slightly different structure view, than is usual in Lisp, give tool for provides a more clear representation of programs and increase their efficiency using modern computers of old architecture.

On the one hand, modern programming languages should effectively use modern machines, but from the other – to give the algorithms clearly express software to facilitate verification of the latter. Strictly functional language has simple in structure, shows a higher severity compared with traditional languages, where there is the assignment operator. This is due, largely, with the way we present of data structures.

Something the choice of data structures of can provide increased efficiency and functional applications on modern computers. In particular, is a problem using the features of the components of the result, because it is important to have elegant solution.

Ones from fundamental properties of programming languages, which enables calculation, clearly describe the language - it is simplest semantics. The great advantage of functional languages is that there are some basic concepts, each of which has a simple semantics. In particular, the semantics of our language understood in terms of the values that are expressions, but not in terms of action sequences and their use. But from a practical point of view it would be fair to conclude that a strictly functional language is very elementary and some of its expansion would significantly increased the efficiency and clarity of certain classes of calculations. Obviously, it is necessary to distinguish between purely syntactic extensions and extensions that require change semantics. Changing the semantics requires caution, because it difficult then to understand already debugged functional programs.

3. STRICTLY FUNCTIONAL LANGUAGE

When it comes to building a strictly functional language, new functions or functional are based on some basic set of primitive features or functions [12]. These include functions:

CAR (X) – it selecting the first element of the list **X**,
CDR (X) – return the remaining elements of list **X** without the first element,
CONS (X, Y) – construction of a new list, where the parameter X is the first element in the list **Y**,
EQ (X, Y) – the predicate is true in the case of equivalent atoms **X** and **Y**.

The presence of these primitive functions and presentation arithmetic operations as functions give the possibility of setting recursive (repeat) functional calls or appeals functions to itself (principle of function composition) can build a fairly substantial functional program [5, 8]. Demonstrate this procedure we can on example building of function the **CONNECTION (X, Y)**, what generates a new list, which will list all the elements of **X** and **Y**. In addition, each argument can be either an atom as a list of arbitrary length. Direct analysis of all possible cases in list X leads to such a functional definition:

CONNECTION (X, Y) = if EQ (X, NIL) then if EQ (Y, NIL) then NIL
another if Y EQ (Y, NIL) then X another
CONS (CAR (X), CONNECTION (CDR (X), Y))

Thus, if X = NIL connect (X, Y) = Y regardless of the value of Y, the definition can be rewritten more optimal way:

CONNECTION (X, Y) = if EQ (X, NIL) then Y
then If EQ (Y, NIL) then X another
CONS (CAR (Y), connection (CDR (X), Y))

If we take into account, that when X = NIL regardless from kind of function **Y** we have **CONS (CAR (X), connect (CDR (X), Y)**, then checking's for **Y** can also be put down and we can write functional definition as:

CONNECTION (X, Y) = if EQ (X, NIL) then Y else
CONS (CAR (X), CONNECTION (CDR (X), Y))

All three versions mean equivalent functions and provide the same result. The first version can be preferred because it clearly lists all possible cases. The third version - is its shortness. But the second and third versions of this tool for connect have different efficiency. The second version avoids calculations in the case **of Y = NIL** by redundant checks when **Y <> NIL**. The third version avoids revisions to Y, but requires rebuild X even when **Y = NI**. Thus, this definition can be done more optimal, particularly using additional sub functions.

Thus, we came to systematically important conclusions after considered constructions of function **CONNECTION(X, Y)**:

a) no other way to solve the problem optimally than continuous improvement achieved,
b) necessary technical instrument for continuous save given knowledge's, them change and improvement.

4. THE USE ADDITIONAL UNDER-FUNCTIONS

The often for building optimal features should be introduced additional functions [13]. Thus, to optimize definition of the function **CONNECTION (X, Y)** previous could be used intermediate function such as **CONNECT(X, Y)**, which connects **X** and **Y** assuming that **Y <> NIL**. Then we arrive at the definition:

CONNECTION (X, Y) = if EQ (X, NIL) then X another CONECT ((Y, NIL)
CONECT (X, Y) = if EQ (X, NIL) then Y
another CONS (CAR (X), CONECT (CDR (X), Y))

Note, that in the previous section we got advantages combined second and third versions thanks of the use function connect

It is generally welcome in a functional programming when the program of building and identifying the main new features are defined in terms of the old. Thus, the functional program consists of sets of sub-functions that are defined through the second one. This feature or functionality that is the purpose of the calculations it is the main program, the root cause of the other, and all other functions are sub-programs.

The choice of sub-functions in the development of the main features is the central problem of structuring programs. Sometimes the standard sub-functions are in themselves, but more frequent occasions when a good selection sub-functions special purpose simplifies the structure of the set of functions in general. To build a well-structured program, we can give one piece of advice: try to constantly improve what is already done. Actually, this is one of the principles of optimal results not only in a functional programming, but also in any other field of knowledge.

5. USE OF ACCUMULATION PARAMETERS

The idea of the method parameters of accumulation is to determine the function of supporting an additional parameter that is used to accumulate the desired result [13].

To illustrate the essence of this programming method to wrap function **ROTATION (X),** this is responsible for the list's elements, possibly empty. For this, we introduce an additional function **ROTATE (X)**, where **X** – a list that is subject to rotation, and **Y** – an additional parameter that accumulates reverse list. We give the following definition:

ROTATION (X) ≡ IF EQ (X, NIL) then Y
ROTATE (CDR (X), CONS (CAR (X), Y))

Because of this feature can determine the function **ROTATION (X))**:

ROTATION (X) = ROTATE (X, NIL)

In beginning we explain how it works, and then describe the algorithm for its construction. When called the function **REVERSE (X, Y)**, y is a list of all accumulated a list of items considered to be the rotated. So if **X** is **NIL**, then y contains all the reverse list, and if x is not NIL, then we can accumulate y **CAR (X)** and then recursively call the sample for processing **CDR (X).** Let us give a table of successive calls function **REVERSE (X, Y)**, that first time by help of functions **ROTION (X)** drawn to the **LIST (A B C D)**.

Table 1. Table of successive calls function REVERSE (X, Y)

X	Y
(A B C D)	NIL
(B C D)	(A)
(C D)	(B A)
(D)	(C B A)
NIL	(D C B A)

In fact, we recorded guessing definition of this function, and then described how this works. In order to apply the conventional method to construct the functions is necessary prove in appearance of results ROTATE(x, y) at an arbitrary y. Let the result of the function rotate (x, y), where x and y are lists, possibly empty, a list of all elements x, are taken in reverse order and are complemented by all elements y in their original order. Thus

ROTATE (X, Y) = CONNECTION (ROTATION(X), Y)

Although this definition is not entirely appropriate because the unknown is the function ROTATE. Yet now we can write the algorithm for constructing functions:

case (1): X = NIL ROTATE (X, Y) = Y
case (2): X = NIL
Let ROTATE(CDR (X), Z) = CONECTION (ROTATION CDR ((X), Z)
then
ROTATE (x, y) = CONECTION (ROTATION (X), Y) =
CONECTION (ROTATION (CDR (x)), CONS (CAR (x), y)) =
= ROTATE (CDR (X), CONS (CAR (X), Y))

The algorithm is faster proved justice above the designated function than the method of its construction, so in the second case, the transformation is quite complex.

The problem of building effective functional programs can be successfully solved by under-function successful recruitment of additional options. Indeed, if we count the number of calls to the designer last version function **ROTATION**, they will be exactly n, if the length of the list is x n. If you compare this number of calls the constructor with a number of n × (n-1) / 2 in the case of the function **ROTATION** (X) without parameters of accumulation, we have significant economy of machine resources.

6. DIRECTIONS OF OPTIMIZATION FUNCTIONAL LANGUAGES

The principal feature of modern computers is that they store calculated values in the memory cells, occasionally replacing their contents or overwriting them. This property is reflected in the design assignment, which is typical for traditional programming languages. There is another aspect of the real machine, through which the traditional language more effective compared to functional programming languages. This concept of access to data structures using indexing. In this connection, functional programs cannot achieve the same effect on modern computers as a program algorithmic programming languages. We can specify at least three outing from this situation:

a) in the unwillingness to put up with some loss of efficiency (in gains in quality and elegance program) refuse functional programs,
b) to develop methods building of functional programs so that their expressiveness and clarity combined with efficiency, comparable with programs written in traditional programming languages,
c) restructure modern computers so as to meet the objectives of interpretation of the functional programs.

From a certain point of view, each of these approaches has the right to exist. Operation appropriation is closely associated with binding value for variable in a functional language. It can be shown that any application where there is a transaction assigning certain variable values can be transformed into functional program. At this functional program with restrictions on its structure can be compiled into the program with the assignment. Thus, one could argue that functional languages do not preclude the effective use of modern computers. But this is only part of the problem because the assignment transaction value of the array type

A[I]:=1
A[I+1]:=A[I]+1

significantly different from operation assignment value of variables. In a strictly functional language is no concept of subtraction sequence of actions and we to behave with arrays as with arrays of integer values. The basic operations on arrays are index values for components and constructing a new array from of old values. If a – values of the array, and – the index, X – value, the record

UPDATE (A, I, X)

means maintaining of array values a and change of **I** - th components on the value of **X**.

And the expression

UPDATE (A I, A [J]), J, A [I])

means replace places of i-th and j-th array elements.

Therefore, the problem of correct interpretation of function correct update is quite complicated taking into account that one and the same value array can have different names. For the operation updating values in the array requires multiple copy these values. The question is: can choose a data structure to fully or partially avoid these copies? Problems arise when used for this purpose simple function as cons, as structures that are components of the cons are not copied, but saved only pointers to them in the structure resulting from cons. But the main advantage is lost today's computers - the possibility of indexing. Indeed, the linear sequential access lists and access time-element is proportional to the number of items and. However, in the array of memory cells simultaneously is access to any item, regardless of position. Arrays can be represented as a binary tree; access time proportional value $\ln_2 n$. However it is incompatible with access time to the unit, that is needed for such elementary statements **a [i]: = x**. Functional language performance on modern computers cannot compete with traditional languages, if efficiency is an absolute requirement. However, applications written with by operators assignment structured values are programs low level. These require considerable effort for their construction and therefore are difficult to correct interpretation.

Increasing the speed and capacity of computer memory nowadays makes available a number of important functional programs applied nature. But qualitative leap in the use of functional languages can be expected only with the advent of new computer architecture that is focused not on the use only of new technology, but her is tied with peculiarities of the language itself, built based on the natural needs of the user.

7. CONCLUSIONS

With using discussed ways and suggested herein of system approach to building applications to optimize complex structured functional programs we can improve of performance, the technical means of realization and form of entry to avoid repetition logical outcome, which is important when working with large volumes of data and database processing of symbolic of information, creation of interpreters other programming languages and recognition of objects complex dynamic structures.

This approach (permanent improvements, from simple to complex, from obvious to unlikely) should be used as in the process of development of new information so and in teaching that will promote a better understanding of material and its efficient use. Creating applications with presentation of new material advisable to implement regardless of content using universal modeling language (umm) [3, 6, 11], that have advanced functional capabilities for creating, storing, permanent updating and presentation of product information.

BIBLIOGRAPHY

[1] Badaev Yu. I., 1999. Theory of functional programming. Movi Common Lisp is Auto Lisp. Kiev, 150.
[2] Edneral V.F., Kryukov A.P., Rodionov A.Ya., 1984. Language of analytical calculations. Publishing House of Moscow State University, 176.
[3] Gamma E., Helm R., Johnson R., Vlissides J., 2010. Wzorce projektowe. Helion Gliwice.
[4] Henderson P., 1983. The functional programming. Application and implementatio. Peace Moscow.
[5] Hüvenen P.E., Slepyan J., 1990. The World of Lisp. In two vol. T.1. Introduction to Lisp language and functional programming. Trans.: Myr, 447.
[6] Jean E. Sammet J.E., 1969. Programming Languages: History and Fundamentals. Prentice-Hall New Yersey.
[7] Kryukov A.P., Rodyonov A.Y., Taranov E.M., Shablyhyn E.M., 1991. Programming language for R-Lisp, Radio and communication, 192.
[8] Maurer W.U., 1972. The programmer's introduction to LISP. London: Macdonald; New York: American Elsevier.
[9] McAllister J., 1987. Artificial Intelligence and PROLOG for Microcomputers. Hodder Arnold London.
[10] Mccarthy J., 1960. Recursive functions of symbolic expressions and their computation by machine, Comm. ACM Vol. 3, P.184-195.
[11] Wrycza S., 2006. UML 2.1, ćwiczenia. Praca zbiorowa, Helion Gliwice.
[12] Zayats V.M., 1999. The summary of the lecture in the course "Functional programming". Lviv, 55.
[13] Zaiats V.M., Zaiats M.M., 2006. Logical and functional programming. Training manual. Publisher Beskyd Bit Lviv, 352.

PODEJŚCIE SYSTEMATYCZNE DO BUDOWANIA PROGRAMÓW FUNKCJONALNYCH

Streszczenie

W pracy są rozważane podstawowe założenia konstrukcji programów funkcjonalnych. Określono główne metody optymalizacji nowych funkcji i funkcjonalności. Autorzy zaproponowali podejście systemowe do budowania i optymalizacji funkcjonalności aplikacji, które mogą być wykorzystane przy tworzeniu systemów rozpoznawania obiektów i zjawisk. Istotą proponowanego sposobu na optymalne rozwiązanie problemu jest ciągła poprawa, bazująca na optymalizacji funkcjonalnej elementów składowych oprogramowania. Rozwinięte oprogramowanie jest zawsze otwarte i może być modyfikowane, udoskonalane lub uzupełniane. Każda z proponowanych w pracy metod, przedstawiona jest jako przykład realizacji w środowisku Lisp.

Słowa kluczowe: program funkcjonalny, język deklaratywny, oprogramowanie zdefiniowane funkcjonalnie, środowisko Lisp, parametry akumulacji

Lista recenzentów prac opublikowanych w 2017 roku
Reviewers list publication published in 2017

Jens Myrup Pedersen (Aalborg University)
Adrian Gligor (Petru Maior University of Tîrgu Mureş)
Cristian Dumitru (Petru Maior University of Tîrgu Mureş)
Łukasz Zabłudowski (Uniwersytet technologiczno-Przyrodniczy)
Mścisław Śrutek (Uniwersytet Technologiczno-Przyrodniczy)
Yarema Savyla (Lviv National University of Ukraine)
Ihor Yavorskyy (Uniwersytet Technologiczno-Przyrodniczy)