TOWARDS GESTURE RECOGNITION IN THREE-DIMENSIONAL SPACE

Łukasz Gadomer

Faculty of Computer Science, Bialystok University of Technology, Białystok, Poland

Abstract: In this work, author describes the continuation of his researches about gesture recognition. The previous varaint of the solution was using plain data and was dependent of the performance velocity. In the described researches author made it speed and position invariant by resolving problem of too long or too short gestures – in a previous solution the user had to decide about gesture duration time before performing, now it is not necessary. He also proposed another data representations, using features computed of recorded data. Previous representation, which assumed storing relative positions between samples, was replaced by transforming each gesture to the axis origin and normalizing. He also tried to connect these two representations – plain data and features – into a single one. All of these new data representations were tested using the SVM classifier, which was judged to be the best for the given problem in the previous work. Each of them was tested using one of four popular SVM kernel functions: linear, polynomial, sigmoid and radial basis function (RBF). All achieved results are presented and compared.

Keywords: data classification, gesture recognition, data features, three dimensional space, speed invariant, position invariant, kinect

1. Introduction

The gesture recognition problem is an issue which many authors are interested in. They treat and understand gestures in a different way and propose different solutions to resolve this problem. Some of these solutions are presented in the current paragraph.

The first approach assumes treating gesture as a movement of a single point (or a set of points) which represents a part of the body (usually a hand). This corresponds with definition which says that gestures are "movements of the arms and hands which are closely synchronized with the flow of speech" [1]. The author's solution, which is described in this publication, is based on this way of treating gestures. Sample

Advances in Computer Science Research, vol. 12, pp. 5-20, 2015.

works, where this approach is used, are [2], [3] and [4]. In all these solutions gestures were collected using accelerometer MEMS (Microelectromechanical System). It is a device which was placed in the user's hand and it was tracking hand's movement in three dimensional space. In [2] authors tried to recognize seven simple gestures. They measured motion in three dimensions, but in fact gestures were two–dimensional. They extracted features from collected data and then performed recognition. Authors of [3] performed their research on 18 different gestures — database was consisting of 3780 instances. They decided to resolve classification problem using Dynamic Time Warping (DTW). Third solution based on accelerometer is described in [4]. Authors were facing gesture recognition problem on 3200 same length (3.40 seconds) gesture instances, grouped into 8 gestures. They also used Dynamic Time Warping algorithm, but hardware accelerated. During the research they tested recognition accuracy in user–dependent and user–independent cases.

Another example of a similar gesture tracking approach is described in [5]. To collect gesture data, authors used finger–worn device called Magic Ring. They described a method of adaptive template adjustment to personalized gesture recognition. In [6] authors designed a solution based on 1$ algorithm which uses Compressed Sensing (CS) and Sparse Representation (SR) methods. Their solution is based on algorithm proposed in [7]. The algorithm is called 1$ to emphasize its low cost and simplicity. The implementation takes about a hundred lines of code. Authors claim it works well even with a single instance of each gesture as a training set. Authors of [8] proposed a solution based on a digital camera, plugged into a computer. The camera images were processed in real time and gesture data extracted from them. The Modified Levenshtein Distance were used as a classification method. Authors performed their research on the dataset consisting of ten digits.

In a solution proposed in this publication author uses Microsoft Kinect device. The same device was used in [9] and [10]. Authors of [9] used gesture recognition as a way of communication with a robot. To test their solution, they performed a research using Dynamic Time Warping algorithm. In [10] gesture recognition process was divided into three stages: modelling, analysis and recognition. The research was performed by authors using Hidden Markov Model.

Sometimes gesture is expressed by a movement of a set of points, each represents different part of the body. The example of recognition such kind of gesture was described in [11]. Authors also used the Kinect device to track user's movements. The points representing different parts of the user's body were used to recognize karate chops.

A different way of understanding gestures assumes treating them as a palm's shape. An example of treating gesture that way is shown in [12]. Author was interpreting gesture as a characteristic arrangement of a palm. Recognition was performed by palm images analysis, which were provided by a digital web camera. A similar approach was presented in [13]. Authors of this publication were studying a possibility of using gestures to interact with a computer game. Similarly to [12], device used to track performed gestures was a digital camera. Gesture recognition was possible thanks to detecting an outline of a palm. In order to check the practical usage of created solution authors tested performing them in a computer game created by themselves. The last example of understanding gestures in a similar way is [14]. In this article authors compared two gesture recognition algorithms: Finger-Earth Moving Distance (FEMD) and Shape Context. They proposed to use their solution in a Sudoku game. To test gesture data they used Microsoft Kinect device.

The last example of understanding gesture concept is sign language. According to [15], each sign can be divided into four parameters: hand shape, position, motion and orientation. Many authors proposed their solutions to track and recognize sign language. One of them is described in [16]. In this publication authors described their way of solving huge search space problem in a large sign vocabulary. Their research was made on 5113 signs and gave results about 95%. Another example is presented in [17]. Authors treat sign language as hand positions and movements. They proposed a sign language recognition method based on a multi–stream HMM technique. Research was performed on 21960 sign language word data. Classification accuracy achieved 70.6% for the same weight of hand position and movement and 75.6% for 0.2 : 0.8 weight proportion, which allowed them to conclude hand movement is more important in sign language.

Gesture recognition is a kind of data classification problem. The point of this issue is creating a solution which is able to track gesture performed by its user and recognize which gesture from the previously learned dataset it was. Author of this publication proposed such solution in [18]. It was able to recognize performed gestures in real–time using one of four selected classifiers. The real–time recognition was position invariant, which means user could stand in a different place of the controller's field of view and in a different distance from it. Recognition was not speed invariant — each set of recognizing samples had the same length. The solution also allowed to test prepared datasets in offline mode. This is the way each classifier was tested and compared with others.

The practical usage of real–time gesture recognition was shown using CAVE3D environment. It is the device which simulates three dimensional space by displaying prepared images on three cube–framed screens. User is placed into this space (between screen walls) and can communicate with it using gestures. Each gesture has assigned action which allows him to, for example, create objects or move.

The continuation of the work described in [18] is presented in this publication. Author concentrated on the strict aspect of gesture recognition and proposed two different data representations in comparison to the one presented in the previous work. Both of them were compared with themselves (and with the combination of them) using the best classifier fitting to the given problem (according to the previous work). He also improved the way of collecting gestures, which made recognition speed invariant.

2. Gesture tracking equipment

Author prepared his own dataset to learn the classifier and perform the research. He obtained gesture data by tracking them using Microsoft Kinect [19]. It is a registering device that has build-in (inter alia) color sensor, depth camera and infrared emitter. All these features allow for recording user's movements in three dimensional space. Color sensor is responsible for width and height recording, depth camera and infrared emitter provide support for the third dimension: the device captures gray scale images; in these images the intensity parameter defines depth, so objects that are located near the camera are represented in a different intensity than objects located far from the camera.

The used Kinect SDK [20] gave possibility of tracking user's skeleton. It is a data structure consisting of characteristic detected user's body points, which coordinates changes in real time (about 30 times in a second) according to current user's body position. Author used this feature to record user's right hand's position changes in time. It was helpful because author preferred to focus on gesture recognition instead of image analysis.

3. Gesture data

3.1 Speed invariance

In [18] authors proposed gesture representation which allowed them to make tracking position invariant. All gestures from the one dataset have defined the common number of samples. According to Kinect's capturing frequency the following samples positions were compared to the previous one. The difference between coordinates of two nieghboring samples was computed separately for each axis and stored in a single vector. It means that for each gesture which length was n samples, there was created $3n+1$ vector.[1] Capturing relative positions of samples instead of direct one allowed to make recognition independent of the place in field of controller's view where the gesture was performed. All vectors from the dataset were also normalized in the recognition process (in the dataset file they were stored unnormalized way), which allowed for comparison of high dimensional points and made recognition independent of the distance between the user and the controller.

This way of capturing gestures was not speed irrelevant. If the recorded gesture was too short (had too small number of samples), program was informing user about that and user had to record another gesture. If the gesture was too long, excessed samples were simply cut from the beginning of the gesture. It means user had to care about the approximate length of gesture he performed. This problem was resolved by changing the way of interpreting recorded gesture. Two ways of dealing with the improper length of recorded gesture were implemented:

- If gesture is too short, program puts additional sample in the middle of the longest distance between two samples in a recorded set. This action is being repeated until the proper number of samples is achieved. It is shown on figure 1, which is ilustrated by the single instance of '*O*' gesture.

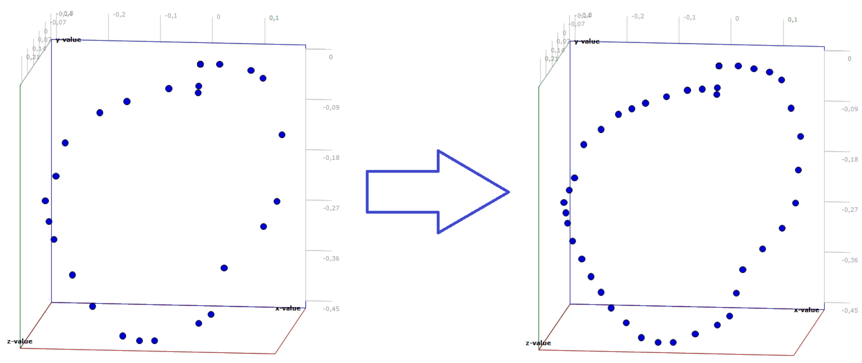

Fig. 1. The way of dealing with too short gestures.

- If gesture is too long, program removes single sample closest to another one in comparison to other neighbour couples in the set. The distance between the

[1] In fact for the gesture with length n there were captured $n+1$ samples to compute differences between coordinates of n couples of neighboring samples.

remaining neighbour of the deleted sample and that sample is added to the new remaining sample's neighbour. This action is repeating until achieving proper number of samples. It is shown on figure 2, which is ilustrated by the single instance of '(' gesture.

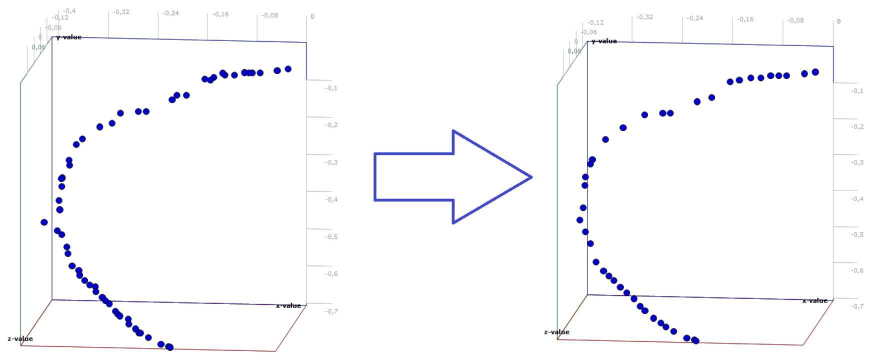

Fig. 2. The way of dealing with too long gestures.

Two proposed ways of dealing with too short or too long gestures makes recognition speed invariant. It means user can perform gesture with any speed — program rebuilds each recording to the same size.

3.2 Data representation

For the purposes of the research described in the following paragraphs recorded dataset were transformed. Performed gestures were written to file consisting of vectors created in the way described before. The idea of new data representation were to make recognition not only speed invariant, but also independent of the different styles of performing the same gesture by different people (for example: different velocity of performing creating gesture parts). What is more, there was an important issue to reduce the number of classifier inputs: for n samples, there were $3n$ inputs, which is a large number. To achieve this goals author decided to extract features from the dataset.

Vectors consisting of relative location to the previous samples were not good source of data to feature extraction. Their values have tendency to oscillate about zero, which was an intentional action for the previous assumptions, but inadequate to the new ideas. To allow for successful feature computation, relative vectors were changed to absolute, but always with their start at the axis origin. This returns recorded gestures to their source representation, but with additional improvement making every gesture start at the same point.

From the dataset prepared that way there were extracted features presented in Table 1. Most of these features were computed to the each of three axis independently, some of them also together for the all axes. Ratio features were computed between cartesian of axes. All in all there were produced 49 features.

Table 1. Features extracted from the dataset.

Feature	Comments
Average	Average value of the each axis and of the all axes.
Standard deviation	Standard deviation of the each axis and of the all axes' values.
Variance	Variance of the each axis and of the all axes' values.
Axis to axis ratio	Ratio between the amplitude of the extreme points of two axes. Computed for the each couple of axes.
Axis to axis correlation	Correlation between the two axes. Computed for the each couple of axes.
Axis to axis covariance	Covariance between the two axes. Computed for the each couple of axes.
Skewness	Skewness of the each axis and of the all axes.
Kurtosis	Kurtosis of the each axis and of the all axes.
Signal magnitude area	Signal magnitude area of the each axis and of the all axes.
Root mean square	Root mean square of the each axis and of the all axes.
Mean deviation	Mean deviation of the each axis and of the all axes.
Interquartile range	Interquartile range of the each axis.
Energy	Energy of the each axis and of the all axes.

Gesture recognition was tested using these features, transformed to absolute values and the same start point dataset and fusion of these two data types, which is further described in the following chapters.

4. Research description

Considering the novel way of recording data the new dataset was prepared. It has the same number and kind of gestures as the dataset tested in [18], but it consists of the new gesture instances. Because of that reason the results between these two researches should not be compared. There are 960 gesture instances in the dataset, divided into 12 different gestures, 80 instances each. These gestures are presented in table 2, which shows also where the gesture performing starts.

There was tested three representations of created dataset, which are mentioned in chapter 3.2. First of them assumed extracting absolute positions of the hand in gesture performance time from the source datafile, transforming it to the beginning at the axis origin and min–max normalizing with the [-1, 1] scope. Each gesture instance had

Table 2. Gestures which belong to dataset.

Gesture shape	Gesture name	Starting point
(	Opening bracket	Top
)	Closing bracket	Top
<	Less than	Top
>	More than	Top
$\wedge$	Caret	Left
\	Backslash	Top
/	Slash	Top
$\vert$	Vertical line	Top
—	Horizontal line	Right
$\sim$	Tilde	Left
O	Circle	Top
8	Eight	Top

117 attributes (distances between 40 samples in 3 axes). The second representation was consisting of features extracted from the previously described one. There were 49 features, which was the number of attributes in this case. The third representation is a fusion of the first and the second one. In this case each gesture instance was described by hand positions and features. It means each gesture instance was described by 166 attributes (summary of two previous representations).

In the previous publication [18] the following classifiers have been tested: SVM [21], Neural Network and Radial Basis Network. SVM classifier gave the best results – it appeard to fit the best to the given problem and it was used to perform classification in this publication. Four kernel functions were tested and compared: linear, polynomial, sigmoidal and radial basis function (RBF). Using each of these functions there were performed the classification of three gesture representations. For each functions and each representation there was a necessity of classifier's parameters optimization. It was performed using 5-fold crossvalidation on the single division of the dataset. It was divided into five equal parts, each of this part was containing the same number of each gesture instances as the other ones. SVM algorithm was learning using four of these parts and testing using remaining one. It was repeated five times, every time another part was the testing one. The result was average of five achieved accuracies. Such operation was repeated many times on different classifier's parameters. The parameters were combined using grid search: for each parameter, there was assumed a range of testing and a step. There was tested all combinations of parameters in assumed ranges. Such optimization was performed for all kernel functions and for three data representations. What is more, after first optimization, there was performed a second one with narrowed ranges to fit dataset the best possible way.

All parameter ranges and test results are presented in tables 5, 6, 7, 8, 9, 10, 11, 12, 13.

After obtaining the best parameters of the classifier's kernel functions for each data representation, the research was performed using them. Like in parameters optimization process, there was used 5-fold crossvalidation, described above. The difference was about the dataset division. In the parameter optimization, there was only one division. In the research, there were used 100 different divisions. Using each of these divisions there was done 5-fold crossvalidation. Then all of one hundred results were averaged, which gave the final accuracy.

5. Results and discussion

The results of the research described in chapter 4. are presented in Table 3.

Table 3. Research results.

Kernel function	Data [%]	Features [%]	Data with features [%]
Linear	95.39	93.36	94.56
Polynomial	95.48	93.37	94.56
Sigmoid	95.35	8.56	8.38
RBF	95.58	88.11	38.71

As we can see, the best results were achieved for plain data representation. Using each tested kernel it was possible to achieve comparable accuracy which was higher than 95%. The best result for this gesture representation gave radial basis function kernel, which was 95.58%. For features representation the results was more differential. Linear and polynomial kernels gave almost the same result (about 93.65%), which was about 2% worse results than for plain data. It is understandable because feature representation has above twice columns less than plain data representation. RBF kernel achieved noticeably worse result: 88.11%. Sigmoid kernel gave the worst accuracy: 8.56% which is almost the same as random. Performing described tests author was unable to adjust this kernel to given problem. The most surprising is the result of data with features representation. It has most columns which means it brings the most information and allows to expect to give the best results of all representations. However, the results were different. For linear and polynomial kernel, the results were comparable (94,55%) and placed between plain data and features representation. For sigmoid and RBF kernels, they were even worse than for features representations. Sigmoid kernel gave results close to random, RBF achieved 38.71%, which is also bad result.

The sample confusion matrix is presented in Table 4. We can observe that the most common reason of missclassification was the similarity of gestures. For example, the gesture ’$<$’ in some cases was recognized as (and vice versa. The way of performing these gestures is quite similar so it could be a bit confusing for the classifier. The same issue was about gestures ’$>$’ and ’)’. Gesture ’$|$’ was missclassified as ’/’ and ’\’ for the same reason. This explains the main reason of mistakes.

Table 4. Sample confusion matrix.

Gest	O	/	\	$\wedge$	$<$	$>$	(	)	8	$\sim$	$\vert$	—
O	99.75	0.00	0.00	0.00	0.25	0.00	0.00	0.00	0.00	0.00	0.00	0.00
/	0.00	99.48	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.52	0.00
\	0.00	0.00	98.68	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.32	0.00
$\wedge$	0.00	0.00	0.13	98.62	0.00	0.00	0.00	0.00	1.25	0.00	0.00	0.00
$<$	0.00	0.00	0.00	0.00	90.45	0.00	9.55	0.00	0.00	0.00	0.00	0.00
$>$	0.00	0.52	0.00	0.00	0.00	92.87	0.00	6.21	0.00	0.00	0.00	0.92
(	0.00	0.00	0.00	0.00	7.93	0.00	89.55	0.00	0.00	0.00	2.52	0.00
)	0.00	0.00	0.00	0.00	0.00	5.47	0.00	92.58	0.00	0.00	1.95	0.00
8	1.31	0.00	0.00	0.00	0.00	0.00	0.00	0.00	98.69	0.00	0.00	0.00
$\sim$	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00	0.00	0.00
$\vert$	0.00	2.68	5.87	0.00	0.00	0.00	1.02	0.64	0.00	0.00	89.80	0.00
—	0.00	2.43	0.00	0.00	0.00	1.51	0.00	0.00	0.00	0.00	0.81	95.25

6. Conclusion

In this paper author described his progress at work on gesture recognition issue. He improved solution developed by himself by making gesture recording and recognizing speed invariant. He also proposed three gesture representations and tested if it is possible to recognize gestures using each of them. To test the recognition accuracy, author used classifier which appeared to be the best for the given problem (according to his previous researches): SVM. He tested four kernel functions: linear, polynomial, sigmoid and radial basis function (RBF). All results were presented and compared.

It turned out that the best results can be achieved using plain data representation. In combination with new gesture tracking method it seems to be the best recognition way for the given problem. However, this gesture representation can bring the problem of too many features, which can make recognition process not efficient enough. The solution can be second proposed representation. It has been tested that for more than twice features less it can give only a bit worse results. When full gesture dataset in plain data recognition way gives satisfying accuracy, it can be beneficial to check feature representation, which allows to reduce recognition time. The third representation was expected to give best accuracy as having the largest number of features, which surprisingly was not proved by performed research. It can be tested for the more complex datasets, which makes the thread for the further researches.

Table 5. Parameters ranges used for the first test in a grid search to find the best ones — dataset containing of plain data.

First test ranges — data					
Kernel function	C	Gamma	Degree	Coef0	Best test result [%]
Linear	1–1000, step 1	X	X	X	95.625
Polynomial	1–50, step 1	0–100, step 1	1–10, step 1	X	96.146
Sigmoid	1–50, step 1	0–100, step 1	X	0–1, step 0.1	95.208
RBF	1–100, step 0-100, step 1	X	X	X	96.146

Table 6. Parameters ranges used for the second test in a grid search to find the best ones — dataset containing of plain data.

Second test ranges — data					
Kernel function	C	Gamma	Degree	Coef0	Best test result [%]
Linear	0.001–1, step 0.001	X	X	X	95.625
Polynomial	46–50, step 0.001	0	3–4, step 1	X	96.146
Sigmoid	1–50, step 1	0–0.04, step 0.002	X	0–0.1, step 0.02	95.208
RBF	81–92, step 0.001	0	X	X	96.146

Table 7. Best parameters achieved in a grid search — dataset containing of plain data.

Best parameters — data				
Kernel function	C	Gamma	Degree	Coef0
Linear	0.71	X	X	X
Polynomial	48	0	3	X
Sigmoid	37	0	X	0
RBF	85	0	X	X

Table 8. Parameters ranges used for the first test in a grid search to find the best ones — dataset containing of features.

First test ranges — features					
Kernel function	C	Gamma	Degree	Coef0	Best test result [%]
Linear	1–1000, step 1	X	X	X	93.333
Polynomial	1–50, step 1	0–100, step 1	1–5, step 1	X	94.167
Sigmoid	1–500, step 1	0–100, step 1	X	0–1, step 0.1	8.646
RBF	1–100, step 0-100, step 1	X	X	X	83.230

Table 9. Parameters ranges used for the second test in a grid search to find the best ones — dataset containing of festures.

Second test ranges — features					
Kernel function	C	Gamma	Degree	Coef0	Best test result [%]
Linear	0.001–1, step 0.001	X	X	X	94.167
Polynomial	10–16, step 0.001	0	1	X	94.167
Sigmoid	10	0	X	5–6, step 0.01	8.646
RBF	0.1–5, step 0.1	0–1, step 0.001	X	X	88.542

Table 10. Best parameters achieved in a grid search — dataset containing of features.

Best parameters — features				
Kernel function	C	Gamma	Degree	Coef0
Linear	0.243	X	X	X
Polynomial	11.92	0	1	X
Sigmoid	10	0	X	5.27
RBF	4.7	0.003	X	X

Table 11. Parameters ranges used for the first test in a grid search to find the best ones — dataset containing of plain data and features.

First test ranges — data and features					
Kernel function	C	Gamma	Degree	Coef0	Best test result [%]
Linear	1–1000, step 1	X	X	X	94.480
Polynomial	1–50, step 1	0–100, step 1	1–10, step 1	X	95.000
Sigmoid	1–50, step 1	0–100, step 1	X	0–1, step 0.1	8.542
RBF	1–100, step 0-100, step 1	X	X	X	40.313

Table 12. Parameters ranges used for the second test in a grid search to find the best ones — dataset containing of plain data and features.

Second test ranges — data and features					
Kernel function	C	Gamma	Degree	Coef0	Best test result [%]
Linear	0.01–10, step 0.01	X	X	X	95.000
Polynomial	10–20, step 0.01	0–0.1, step 0.001	1	X	95.000
Sigmoid	0–50, step 1	0–0.04, step 0.02	X	0–0.1, step 0.02	8.542
RBF	1–100, step 1	38.5–39.5, step 0.1	X	X	40.521

Table 13. Best parameters achieved in a grid search — dataset containing of plain data and features.

Best parameters — data and features				
Kernel function	C	Gamma	Degree	Coef0
Linear	0.08	X	X	X
Polynomial	14	0	1	X
Sigmoid	8	5	X	0
RBF	27	38.6	X	X

References

[1] D. McNeill. *Gesture and Thought*. University of Chichago Press, Chicago, USA, 2007.

[2] R. Xu, S. Zhou, and W. J. Li. Mems accelerometer based nonspecific-user hand gesture recognition. *Sensors Journal, IEEE*, 12(5):1166–1173, May 2012.

[3] A. Akl, C. Feng, and S. Valaee. A novel accelerometer-based gesture recognition system. *Signal Processing, IEEE Transactions on*, 59(12):6197–6205, Dec 2011.

[4] S. M. A Hussain, and A. B. M. H. Rashid. User independent hand gesture recognition by accelerated dtw. In *Informatics, Electronics Vision (ICIEV), 2012 International Conference on*, pages 1033–1037, May 2012.

[5] Y. Zhou, D. Saito, and L. Jing. Adaptive template adjustment for personalized gesture recognition based on a finger-worn device. In *Awareness Science and Technology and Ubi-Media Computing (iCAST-UMEDIA), 2013 International Joint Conference on*, pages 610–614, Nov 2013.

[6] A. Boyali, and M. Kavakli. A robust gesture recognition algorithm based on sparse representation, random projections and compressed sensing. In *Industrial Electronics and Applications (ICIEA), 2012 7th IEEE Conference on*, pages 243–249, July 2012.

[7] J. O. Wobbrock, A. D. Wilson, and Y. Li. Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes. In *ACM Symposium on User Interface Software and Technology (UIST '07). Newport, Rhode Island*, pages 159–168, July 2007.

[8] C. Nyirarugira, H.-R. Choi, J. Kim, M. Hayes, and T. Kim. Modified levenshtein distance for real-time gesture recognition. In *Image and Signal Processing (CISP), 2013 6th International Congress on*, volume 2, pages 974–979, Dec 2013.

[9] S. Bodiroza, G. Doisy, and V. V. Hafner. Position-invariant, real-time gesture recognition based on dynamic time warping. In *Human-Robot Interaction (HRI), 2013 8th ACM/IEEE International Conference on*, pages 87–88, March 2013.

[10] Y. Wang, C. Yang, X. Wu, S. Xu, and H. Li. Kinect based dynamic hand gesture recognition algorithm research. In *Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2012 4th International Conference on*, volume 1, pages 274–279, Aug 2012.

[11] T. Hachaj, M. R. Ogiela, and M. Piekarczyk. Image processing and communications challenge 5. In *Real-Time Recognition of Selected Karate Techniquies Using GDL Approach*, pages 99–106, 2014.

[12] M. Panwar. Hand gesture recognition based on shape parameters. In *Computing, Communication and Applications (ICCCA), 2012 International Conference on*, pages 1–6, Feb 2012.

[13] S. S. Rautaray, and A. Agrawal. Interaction with virtual game through hand gesture recognition. In *Multimedia, Signal Processing and Communication Technologies (IMPACT), 2011 International Conference on*, pages 244–247, Dec 2011.

[14] Z. Ren, J. Meng, and J. Yuan. Depth camera based hand gesture recognition and its applications in human-computer-interaction. In *Information, Communications and Signal Processing (ICICS) 2011 8th International Conference on*, pages 1–5, Dec 2011.

[15] W. Gao, J. Ma, S. Shan, X. Chen, W. Zheng, H. Zhang, J. Yan and J. Wu. Handtalker: A multimodal dialog system using sign language and 3-d virtual human. In Tieniu Tan, Yuanchun Shi, and Wen Gao, editors, *ICMI*, volume 1948 of *Lecture Notes in Computer Science*, pages 564–571. Springer, 2000.

[16] G. Fang, W. Gao, and D. Zhao. Large vocabulary sign language recognition based on fuzzy decision trees. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 34(3):305–314, May 2004.

[17] M. Maebatake, I. Suzuki, M. Nishida, Y. Horiuchi and S. Kuroiwa. Sign language recognition based on position and movement using multi-stream hmm. In *Universal Communication, 2008. ISUC '08. Second International Symposium on*, pages 478–481, Dec 2008.

[18] Ł. Gadomer, M. Skoczylas. Real time gesture recognition using selected classifiers. *Architecturae et Atribus*, 6(1):14–18, 2014.

[19] Microsoft. Kinect for Windows. `http://www.microsoft.com/en-us/kinectforwindows/`, 30.03.2015.

[20] Microsoft. Kinect for Windows SDK v1.7. `http://www.microsoft.com/en-us/download/details.aspx?id=36996`, 08.05.2014.

[21] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

Łukasz Gadomer

ROZPOZNAWANIE GESTÓW W PRZESTRZENI TRÓJWYMIAROWEJ

Streszczenie W niniejszym artykule autor opisał kontynuację swoich badań dotyczących rozpoznawania gestów. Ulepszył on stworzone przez siebie rozwiązanie w taki sposób, aby nagrywanie i rozpoznawanie gestów było niezależne od szybkości ich wykonywania, a co za tym idzie — ich zróżnicowanej długości. Zaproponował on także inne reprezentacje danych, za pomocą których wyrażany jest stworzony zbiór gestów. Wcześniejsze rozwiązanie, opierające się na przechowywaniu relatywnego położenia dłoni w stosunku do poprzedniej zarejestrowanej próbki (poprzedniego położenia), zastąpione zostało sprowadzeniem gestu do początku układu współrzędnych i zastąpieniem wartości relatywnych absolutnymi, a następnie ich normalizacją. Z tak przygotowanego zbioru gestów obliczone zostały cechy stanowiące drugą zaproponowaną reprezentację danych. Trzecia reprezentacja stanowi połączenie dwóch poprzednich: zawiera jednocześnie bezpośrednie wartości wyrażające ruch dłoni, jak i obliczone na podstawie jego cechy. Wszystkie trzy reprezentacje zostały przetestowane przy pomocy klasyfikatora, który okazał się najlepszy dla zadanego problemu podczas przeprowadzania wcześniejszych badań: SVM. Porównano, jak z zadanym problemem radzą sobie cztery popularne funkcje jądra: liniowa, wielomianowa, sigmoidalna i radialna. Otrzymane wyniki zostały przedstawione, porównane i omówione.

Słowa kluczowe: klasyfikacja danych, rozpoznawanie gestów, cechy, przestrzeń trójwymiarowa, niezależne od prędkości, niezależne od położenia, kinect