Recognition of Yoga Poses through an Interactive System with Kinect device

Edwin W. Trejo [1], Peijiang Yuan[2].
School of Mechanical Engineering and Automation
Beihang University
Beijing, China
e-mail: [1] e.trejo@pucp.pe, 2 itr@buaa.edu.cn

***Abstract*—In daily life, Yoga has become a well-known discipline around the world that keep people in good physical and mental health. As well, gesture recognition is a field of investigation that takes great importance for the self-training of various sports using acquisition techniques such as Kinect device. This research proposes an interactive system capable of recognizing 6 poses for learning Yoga that can track up to 6 people at the same time. It is also integrated with command voices to visualize the instructions and pictures about the poses to be performance for the user. In order to get a strong database for recognition, the system used Adaboost algorithm though the software development kit specially for Kinect. All data was trained by an expert yoga trainer and final database showed above 94.78% as maximum value for poses analyzed in terms of accuracy.**

Keywords-gesture recognition; Kinect; Yoga; human computer interaction; supervised learning

I. INTRODUCTION

THESE last years, people work with more pressure and they suffer from stress more quickly than before. Even children also suffer because they spend too much time on smartphones and tablets. Recently, Yoga has become very significant and well known around the world. Many reports indicate that this is because of the benefits it offers. Strength, endurance and flexibility are some of the advantages that in developing them help to maintain the balance of body and spirit at the same time.

Currently several sports disciplines and physical activities are being equipped with technological devices so you can get better performance. There are different opinions on whether it is good to merge technology with nature, since Yoga is not only doing exercises but has a philosophical thought that goes back many years ago. This research does not seek to discern what is the best definition for this discipline, it is about providing another tool for people who practice yoga more dynamically and with greater precision when detecting postures.

With the introduction of low-cost depth cameras, new methods for body tracking and gesture recognition were implemented. Furthermore, it has allowed the development of research in various fields, for instance sign language word recognition [1], balance rehabilitation [2], face recognition [3] and gesture interaction [4]. One of these cameras is the Microsoft Kinect sensor that since the launch of its second generation in 2014 has facilitated improvements for hand and face gestures recognition. In comparison with its first version, Kinect v2 increases the accuracy and precision in general, although it requires more computational time to build more complex models [5]. Basically, the Kinect camera is a motion detection input device that works with 3 data streams (depth, color and body tracking). In addition, it includes advanced audio capabilities with the integration of Windows speech recognition and a software development kit (SDK 2.0) [6]. All these features allow to develop an integrated Human Computer Interaction (HCI) system in which takes place as an open-ended dialog between the user and the computer.

This paper proposes a method to recognize 6 common Yoga poses by a Kinect sensor. The recognition of the poses is done through the use of Adaboost algorithm, which in the beginning is processed training data to obtain a final database with high accuracy. In addition, it is integrated into an interactive system which contains voice recognition to execute certain voice commands predefined by the system.

II. RELATED WORK

A. Pose and Gesture Recognition Methods

Over the last decade, human motion recognition and posture estimation have become a popular research topic. There are different methods for pose recognition. Zhiguang Liu et al. proposed to use Kinect for posture reconstruction based on a local mixture of Gaussian Process models [7]. They applied the Gaussian Process algorithm to learn the correlation between the Kinect data and the motion capture data to predict the deviation of the Kinect data since they found that the noise produced by the captured postures was not random but followed patterns underlying. Nevertheless, this machine learning is not competent in systems that require a large database. For this reason, they optimized this model through their method.

Other techniques for pose recognition is to use Neural Network algorithm. Salvatore Sorce et al. presented a method to recognize whether a hand is open or closed base on this machine learning [8]. They use hand segmentation with the RGB frames from Kinect camera and then use three different approaches as inputs for the Neural classifier. According to their experiments and discussion, their approach that include hand mask with edge do not get good results and need more independent algorithms for edge extraction.

The last method reviewed was inverse Kinematics and gesture pattern recognition using Hidden Markov Model [9]. This research, classifies all the data acquired by the Kinect into one degree joint and two degree joint. After that, processing the learning and recognition of HMM summarize

978-1-5386-7370-6/18/$31.00 ©2018 IEEE

the big data from XYZ coordinates of each joint into a more compact data. Although, it is concluded two degree joint do not get good results because of randomized value appeared when the body position is not aligned with Kinect.

B. Yoga Pose Detection

Eyes-Free Yoga is an exergame using depth cameras for blind and low vision exercise [10]. Users can interact with a yoga instructor and receive audio-based instructions for six standing yoga poses. Kinect 1 was used for this project and can track 20 body joints in comparison with its successor version that can track 25 joints. Its method to recognize yoga poses is to calculate the different body angles using the Law of Cosines. Besides, through of an average of 11 rules for each pose, the priority of adjustments in the human body is determined. It is in this way, how the feedback of instructions for the correction of the body is selected.

YogaST is self-training system aims at instructing the user/practitioner to achieve yoga posture correctly and preventing injury caused by improper postures [11]. They used two Kinect 1device with perpendicular viewing directions to obtain the body map. Then, for pose recognition, applied image processing to get contour map and convolved with line masks. With those data, system extracts the axis body user to compare and analyze the correct posture. The main difficulty is when there is overlap of the body parts since segmentation cannot occur correctly.

Xin Jin et al. developed a virtual personal trainer via the Kinect sensor [12]. Principally, they recorded standard exercising actions and stored in a database. When user interacts with the program, three parameters are shown in the body stream which are user body frame, coach body frame and arrow guides to make corrections. There is also a fitness scores that displays the result of performance. This score is processed by Kinect SDK and Dynamic Space-time Warping to finally be calculated by 3D distances between the joints of the user action and the standard action.

Microsoft Kinect provides an excellent tracking application development interface that enables human activity analysis like learning Yoga. Although, there are many approaches and solutions to this research, our system aims to bring a more robust method of recognizing gestures and multitarget detecting users pose.

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$
Initialize $D_1(i) = 1/m$.
For $t = 1, \ldots, T$:

- Train base learner using distribution D_t.
- Get base classifier $h_t : X \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.
- Update:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final classifier:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

Figure 1. Pseudocode of Ada-Boost algorithm.

III. System Description

A. Ada-Boost Algorithm

AdaBoost is an ensemble method that generates strong classifier from a number of weak classifiers. Generally, it is used to get binary classification results. At the beginning, this algorithm predicts the original data set with the equal weight as early learning. After each iteration, the misclassified data is given a high weighting based on the observations obtained when applying the previous classification rules [13]. In this way, it continues to add learners until a strong classifier is reached in order to get high accuracy to the model. The pseudocode of AdaBoost algorithm is shown in Figure 1.

B. Training and Recognition

For this study, 6 Asanas poses based on Hatha Yoga were selected [14]. Each of them has a different degree of difficulty and characteristics such as symmetry, seated or not and support on the ground with the hands in order to analyze the confidence value in the recognition of the system (See Fig.2).

Figure 2. The six Yoga poses used for the recognition of the system. (a) Dragon flow. (b) Gate pose. (c) Tiger flow. (d) Tree pose. (e) Triangle pose. (f) Warrior flow. For more details, the reader is referred to [14].

Kinect for windows SDK v2 has 2 important tools to build the database of the six yoga poses mentioned above, which are Kinect Studio 2.0 (See Figure3) and Visual Gesture Builder (VGB) [15]. The process of building the database was divided into 2 stages: Training data and Analysis data. First, with the

help of an expert yoga trainer, 5 clips were recorded for each Yoga pose with a framerate of 30 frames/sec.

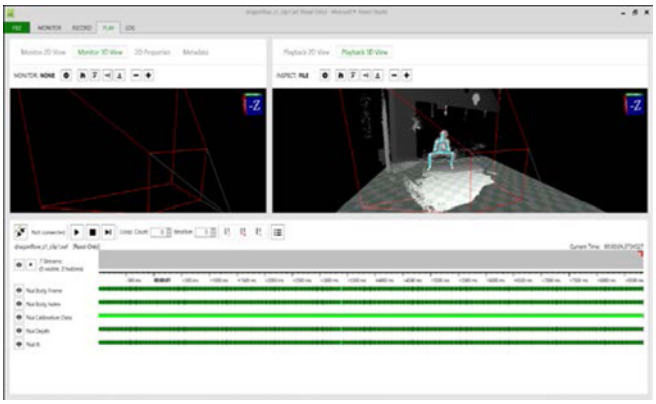

Figure 3. Recording of Dragon pose in KS.

It took 2 training sessions in which a maximum of 10 clips were obtained for each Yoga pose in different environments. After that, the first 5 clips were included in the training data to be processed by the Adaboost algorithm. In that way, the first database was obtained and we passed to the second stage of analysis data. This last data includes the other 5 clips and was processed by one condition. If the sample data gets more than 0.9 confidence value, then this clip was included in Training Data and all is reprocessed again as a new one until the final database is created for the all yoga poses. The flowchart for building the database is shown in Figure 4.

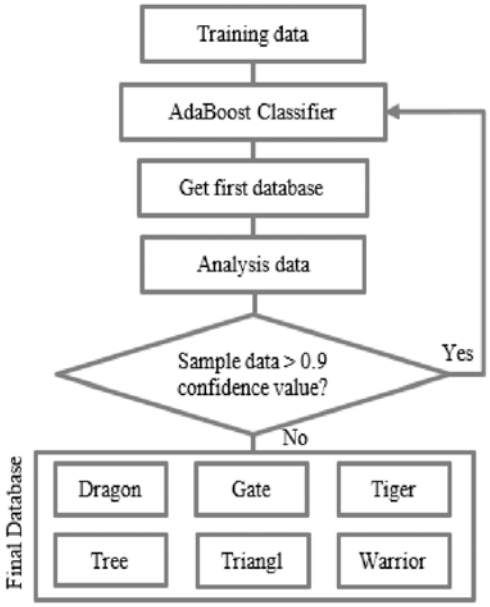

Figure 4. Flowchart of recognition gestures for database.

Also, when creating the database, it was generated a log file that indicates the 10 weak classifiers by Adaboost algorithm. These weak classifiers are conditioned by parameters such as angles, speed and angle velocity. However, our study focuses on angles between joints since it recognizes discrete gestures and do not need to identify parameters related to movements like continuous gestures. VGB trained all data using skeleton-body frame.

In Table I, the 3 strong classifiers were obtained after filtering other features and only finding the angles. Each classifier has 2 parameters: fValue and alpha, that means the condition value to be recognized and weight classifier respectively. For example, a strong classifier for dragon pose can be interpreted as one condition. If the angle between KneeLeft, SpineMid and KneeRight joints is greater than or equal to 104.00f then DragonPoseDetection is true. Similarly, the other poses can be evaluated by following the classifiers achieved.

TABLE I: TOP 3 FINAL CLASSIFIERS FOR EACH YOGA ASANA OBTAINED BY ADABOOST ALGORITHM.

DRAGON	Angles(KneeLeft, SpineMid, KneeRight) using inferred joints, fValue >= 104.0000, alpha = 1.7790
	Angles(Head, ShoulderLeft, KneeLeft) using inferred joints, fValue >= 170.0000, alpha = 0.6235
	Angles(KneeLeft, SpineMid, KneeRight) using inferred joints, fValue >= 108.0000, alpha = 0.4674
GATE	Angles(WristLeft, SpineMid, KneeLeft) rejecting inferred joints, fValue >= 148.0000, alpha = 0.9213
	Angles(HandLeft, Head, HandRight) using inferred joints, fValue >= 160.0000, alpha = 0.6603
	Angles(Head, ShoulderLeft, Neck) rejecting inferred joints, fValue < 12.0000, alpha = 0.5970
TIGER	Angles(KneeLeft, SpineMid, KneeRight) using inferred joints, fValue >= 50.0000, alpha = 1.2677
	Angles(WristLeft, SpineMid, KneeLeft) using inferred joints, fValue >= 84.0000, alpha = 0.5308
	Angles(SpineMid, Head, SpineBase) rejecting inferred joints, fValue >= 14.00000, alpha = 0.4507
TREE	Angles(Head, ShoulderRight, HandRight) using inferred joints, fValue < 88.0000, alpha = 1.5166
	Angles(Head, ShoulderRight, AnkleRight) using inferred joints, fValue < 142.0000, alpha = 1.1545
	Angles(ShoulderLeft, SpineShoulder, ShoulderRight) using inferred joints, fValue >= 148.0000, alpha = 0.5620
TRIANGLE	Angles(Head, ShoulderLeft, FootLeft) rejecting inferred joints, fValue < 98.0000, alpha = 1.0942
	Angles(HandLeft, Head, HandRight) using inferred joints, fValue >= 144.0000, alpha = 0.6768
	Angles(Head, ShoulderLeft, SpineBase) rejecting inferred joints, fValue >= 148.0000, alpha = 0.5751
WARRIOR	Angles(KneeLeft, SpineMid, KneeRight) using inferred joints, fValue >= 92.0000, alpha = 1.4188
	Angles(Head, ShoulderRight, ElbowRight) using inferred joints, fValue < 106.0000, alpha = 0.8045
	Angles(WristLeft, Head, WristRight) using inferred joints, fValue >= 154.0000, alpha = 0.5242

C. *System Design*

This interactive system was developed in Visual Studio as WPF [16]. The solution explorer is shown in Figure 5, which contains 3 carpets for reading Database, Images and Sounds. Additionally, it includes 4 classes called GestureDetector, GestureResultView, KinectAudioStream and KinectBodyView. There is also an InputDialog.xaml to show some messages when the interactive system needs to communicate with the user and the MainWindow in which is the main structure of the program. To summarize the meaning of these classes, GestureDetector has the function to listen for

VisualGestureBuilderFrame events from the service and updates the associated GestureResultView object with the latest results for each Yoga poses. When this is completed, GestureResultView stores the discrete gesture results for the GestureDetector and all its properties are stored and updated for display in the main window. Concerning the KinectAudioStream class, it only supports the speech recognition specially for Kinect. And the last class KinectBodyView is used to visualize the Kinect Body stream for display in the main window.

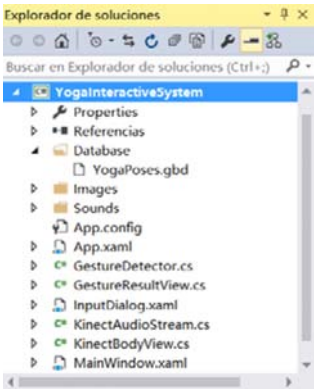

Figure 5. Solution explorer for the interactive system.

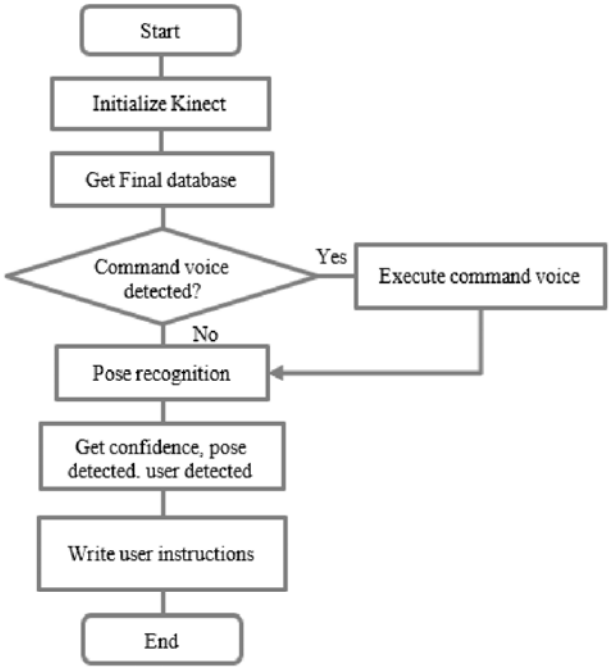

Figure 6. Flowchart of Yoga interactive system.

The next Figure 6 explains the flow diagram of our Yoga interactive system. First the initialization of the Kinect is done by enabling the 3 streams (colorStream, depthStream and bodyStream) and declaring the global parameters that are executed in the program. Then, the system proceeds to read the database generated with the training data for the 6 Yoga poses proposed. Afterward, the speech recognition condition is executed to determine if the program detected some voice command by the user. If it is affirmative, the instruction of said command voice is performed. Otherwise, it changes to the process of recognition poses. For this type of recognition, it was used Microsoft Speech SDK v11.0 [17] due to this model was designed exclusively for Kinect. Returning to poses recognition, the interactive system processes the arriving frame and compare with the final database to provide 3 important parameters that can be visualized in the main window which are confidence value, name of pose detected and user detected. Finally, improvement instructions are displayed for the user that are previously made by the Yoga expert so it does not provide detailed information of the pose made by the user.

D. System Operation

The initial user interface (UI) developed for this research is shown in Figure 6. UI is made to the right side by a display showing BodyStream user, while on the left side ColorStream is displayed to see the user's performance executed in each pose. On this side, there is also a welcome message and a text box that indicates the yoga instructions. In the lower left, there is a small box containing command voices which can be executed in the system for more interaction. These command voices are Next, Previous, Music On, Music Off and Ok. In the lower part there are also 6 avatars that display the detection parameters specified previously in the flow diagram. To finish, there is a status text that indicates if Kinect is running correctly or not connected.

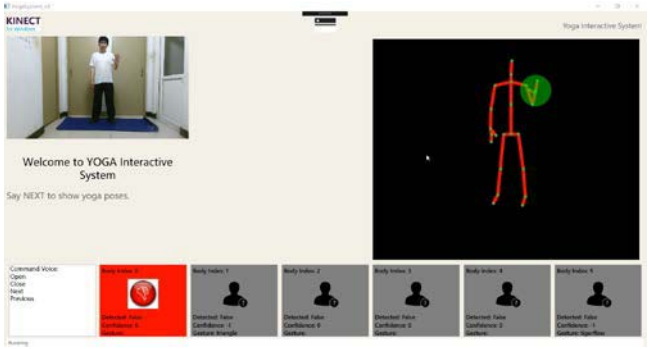

Figure 6. Initial UI of Yoga interactive system.

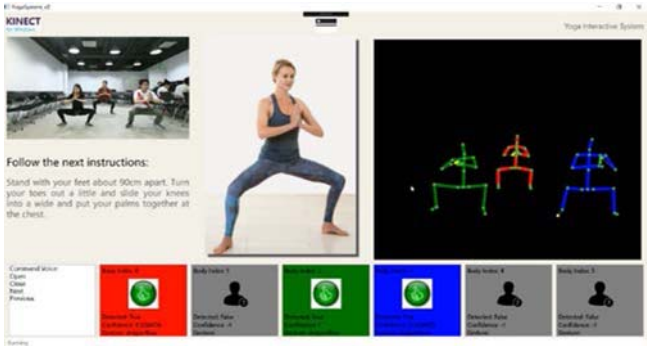

Figure 7. Multiuser detecting dragon Asana.

The operation of the system starts when the user executes the command voice Next, which has as function to put the

following yoga pose image. In case user wants to visualize the previous image, proceed to mention the voice command Previous. While images are displayed, the instructions also appear for proper performance. After pose detection, the user avatar will change to a green image along with the name of the pose that has been detected and its confidence level. Otherwise, it will continue to show the red image in the avatar. For example, Figure 7 represents the UI detecting dragon Asana for 3 users at the same time. An optional function in the program is to play a music in the background, mentioning the command voice Music on.

IV. Experiment

For this experiment, we trained 3 databases in order to analyze the accuracy of interactive system. First database, only the clips of the yoga coach of the first session were considered. For the second database were included clips recorded in the second session performed by yoga trainer. The third database was created with a mix clips of both sessions but limited to 5 clips. In order to develop the databases, it was necessary a tagging process for each Yoga pose (See Figure 8). This step consisted in tagging the frames with positive results as true value detection in the Visual Gesture Builder and following the patterns advised by the trainer for each pose.

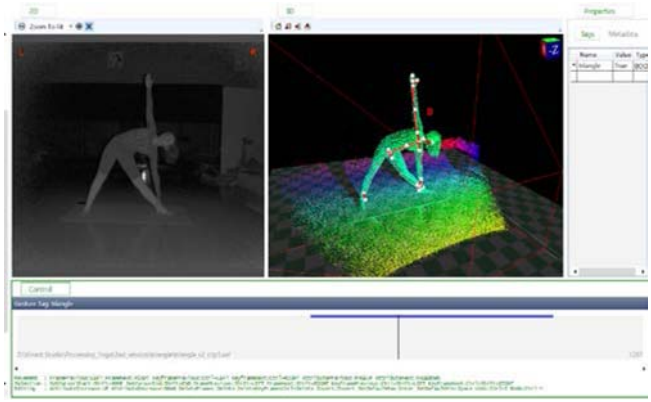

Figure 8. Tagging Triangle pose performanced by yoga trainer.

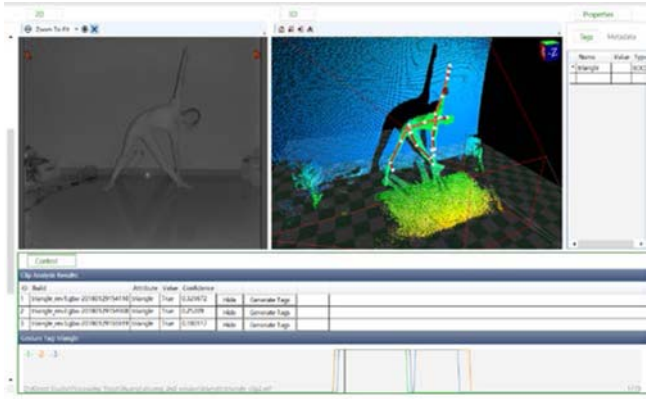

Figure 9. Analysis of Triangle pose by amateur subject.

Apart from that, it was used the analyze part in VGB to find which frames are correctly detected as true value according to database made by yoga trainer. For this reason, it was recorded 3 clips for each pose by an amateur subject. Figure 9 shows an example of data analysis for Triangle pose. In the control bar are specified the confidence value at the current frame as well as a timeline is presented as a step function for the 3 databases. All the results were exported as log file.

V. Results

After obtaining the detected frames, the system was evaluated in terms of accuracy. Table II shows the confusion matrix that reports the number of true negative, false positive, true positive and false negative frames respectively. Likewise, the total number of frames analyzed for each Yoga pose and the results obtained for the 3 databases represented by the nomenclatures D1, D2, D3 are mentioned.

TABLE II: CONFUSION MATRIX FOR THE 6 YOGA POSES CORRESPONDING TO A RECORDED CLIP FOR THE ANALYSIS OF THE 3 DATABASES

Yoga Pose		Total Frames	True Positive	False Positive	True Negative	False Negative
Dragon	D1	372	142	0	216	14
	D2	372	141	0	216	15
	D3	372	139	0	222	11
Gate	D1	378	91	4	254	29
	D2	378	83	12	276	7
	D3	378	106	8	229	35
Tiger	D1	307	29	0	250	28
	D2	307	29	0	262	16
	D3	307	29	0	261	17
Tree	D1	267	8	10	249	0
	D2	267	13	5	242	7
	D3	267	18	0	242	7
Triangle	D1	322	20	2	282	18
	D2	322	63	2	249	8
	D3	322	61	4	257	0
Warrior	D1	249	129	0	92	28
	D2	249	132	0	98	19
	D3	249	134	2	102	11

Then, the accuracy of the algorithm presented was calculated. Table III proves the high accuracy achieved in each database constructed for each of the poses. Green painted cells correspond to the maximum values and red correspond to the minimum values. In general, the results show the following important points.

- Database 3 obtains the best values of accuracy above 94.78% for all poses.
- The first recorded clip for data analysis shows better performance while the third clip presented lower performance since 3 of the poses are in this group.

- For database 3, the minimum value found is 80.6% and the maximum is 98.76% that correspond to Tree pose and Triangle pose respectively.
- Dragon pose gets the best results compared to the other poses. While Tiger and Tree get the lower accuracy values.

TABLE III: ACCURACY ANALYSIS FOR THE 6 YOGA POSES

	CLIP 1			CLIP 2			CLIP 3		
ACCURACY (%)	DATABASE 1	DATABASE 2	DATABASE 3	DATABASE 1	DATABASE 2	DATABASE 3	DATABASE 1	DATABASE 2	DATABASE 3
Dragon	96.24	95.97	97.04	95.06	94.68	95.44	92.15	95.04	95.87
Gate	91.27	94.97	88.62	96.77	97.04	98.92	90.81	96.1	98.05
Tiger	90.88	94.79	94.46	95.77	91.54	96.07	65.76	89.9	87.93
Tree	96.25	95.51	97.38	75.12	98.01	80.6	93.3	96.65	88.52
Triangle	93.79	96.89	98.76	94.6	97.16	96.31	83.9	97.6	93.49
Warrior	88.76	92.37	94.78	96.63	92.46	93.26	90.32	91.4	88.71

VI. CONCLUSION

This paper has presented an interactive system that uses Kinect v2 for 6 Yoga poses recognition with command voices to visualize the instructions and pictures about the poses to be performance. We aim that this system be used for the development of assisted Yoga activity that can improve the user's performance. The results reported in this investigation lead to a high degree of recognition. For this reason, for future work we intend to include other sports disciplines such as Taichi or martial arts that require a greater analysis in recognition patterns since the movements are continuous. In the same way, expand the system to more interaction options to store the data of the user's performance.

ACKNOWLEDGEMENT

The author greatly appreciates the contributions of Ms. Yang ShuangShuang and Yoga trainer Li Ran for their participation in developing databases.

REFERENCES

[1] T. Kodama, T. Koyama, and T. Saitoh, "Kinect sensor based sign language word recognition by mutli-stream HMM," *2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, 2017.

[2] L. Beaulieu, "Balance Rehabilitation using Xbox Kinect among an Elderly Population: A Pilot Study," *Journal of Novel Physiotherapies*, vol. 05, no. 02, 2015.

[3] B. Y. Li, A. S. Mian, W. Liu, and A. Krishna, "Using Kinect for face recognition under varying poses, expressions, illumination and disguise," *2013 IEEE Workshop on Applications of Computer Vision (WACV)*, 2013.

[4] H. Cook, Q. V. Nguyen, S. Simoff, T. Trescak, and D. Preston, "A Close-Range Gesture Interaction with Kinect," *2015 Big Data Visual Analytics (BDVA)*, 2015.

[5] A. D. Calin, "Variation of pose and gesture recognition accuracy using two kinect versions," *2016 International Symposium on Innovations in Intelligent Systems and Applications (INISTA)*, 2016.

[6] Kinect for Windows SDK 2.0 (2014). Microsoft [Online]. Available: https://www.microsoft.com/en-us/download/details.aspx?id=44561

[7] Z. Liu, L. Zhou, H. Leung, and H. P. Shum, "Kinect Posture Reconstruction Based on a Local Mixture of Gaussian Process Models," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 11, pp. 2437–2450, Jan. 2016.

[8] S. Sorce, V. Gentile, and A. Gentile, "Real-Time Hand Pose Recognition Based on a Neural Network Using Microsoft Kinect," *2013 Eighth International Conference on Broadband and Wireless Computing, Communication and Applications*, 2013.

[9] Z. A. Ulfah, A. I. Wuryandari, and Y. Priyana, "Inverse kinematics and gesture pattern recognition using Hidden Markov Model on BeatMe! project: Traditional dance digitalization," *2015 International Conference on Electrical Engineering and Informatics (ICEEI)*, 2015.

[10] K. Rector, C. L. Bennett, and J. A. Kientz, "Eyes-free yoga," *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility - ASSETS 13*, 2013.

[11] H.-T. Chen, Y.-Z. He, C.-L. Chou, S.-Y. Lee, B.-S. P. Lin, and J.-Y. Yu, "Computer-assisted self-training system for sports exercise using kinects," *2013 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, 2013.

[12] X. Jin, Y. Yao, Q. Jiang, X. Huang, J. Zhang, and X. Zhang, "Virtual Personal Trainer via the Kinect Sensor," *16th IEEE International Conference on Communication Technology (ICCT)*, 2015.

[13] S. Hwang, J. Na, S. Park, and J. Baek, "Ada-Boostbased Gesture Recognition using Time Interval Window," *Proceedings of the 2015 International Conference on Electrical, Automation and Mechanical Engineering*, 2015.

[14] C. Brown "*The modern yoga bible: the definitive guide to yoga today"*, Godsfield, pp. 32, 44, 46, 94, 103, 114, 2017.

[15] Channel 9. (2018). *Custom Gestures End to End with Kinect and Visual Gesture Builder*. [online] Available at: https://channel9.msdn.com/Blogs/k4wdev/Custom-Gestures-End-to-End-with-Kinect-and-Visual-Gesture-Builder [Accessed 2 Sep. 2017].

[16] Visual studio. (2015). Microsoft.

[17] Microsoft Speech Platform version 11 (2012). Microsoft [Online]. Available: https://www.microsoft.com/en-us/download/details.aspx?id=27226