2018 3rd International Conference on Advanced Robotics and Mechatronics (ICARM)

Recognition of Yoga poses through an interactive system with Kinect based on confidence value

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

***Abstract*—Nowadays, the recognition of poses is a field of investigation that takes incredible significance for oneself preparing in different sports. Kinect offers a low-cost solution for the recognition of Yoga poses due to body tracking and depth sensor. In this research, we propose an interactive system for perceiving a few postures for learning Yoga that will be characterized by a level of trouble and coordinated with command voices to envision the guidelines and pictures about the stances to be execution. Likewise, posture correction instructions will be displayed for the user in real time made by an expert yoga trainer. Besides, the recognition algorithm is based on Adaboost algorithm in order to get a robust database for detecting 6 Asana Yoga poses. All data were obtained and analyzed according to the confidence which showed a maximum average value of 92%.**

***Index Terms* — Gesture recognition, Kinect, Sports training, Yoga, Human computer interaction, Machine learning.**

I. INTRODUCTION

At present, we are always in constant movement, always exposed to the noise of information, visual congestion and the media. So, we never have time to be with ourselves or even to really relate to others and to the nature of the world around us. During the last decade, Yoga has improved the health of those who practice it, thanks to its multiple benefits. The yoga poses do not stop surprising because of the physical, mental and spiritual benefits that are obtained in yoga classes. Its practice guarantees a firm body, a stable mind and a benevolent spirit.

With the inclusion of low-cost depth cameras, new techniques for body tracking and gesture recognition were implemented. Furthermore, it has permitted the advancement of research in different fields, for instance sign language word recognition [1], balance rehabilitation [2], face recognition [3] and gesture interaction [4].

Microsoft Kinect v2 was launched to the market in 2014. This second-generation depth camera has achieved important developments for hand and face gestures recognition. One research made by Calin, stressed that Kinect v2 in comparison with its first version, increased 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 incorporates advanced audio capabilities with the integration of Windows speech recognition. All these features allow to develop an integrated Human Computer Interaction (HCI) system in which happens as an open-ended dialog between the user and the computer.

This paper proposes a technique to perceive 6 regular Yoga poses by a Kinect sensor. Initially, the Adaboost algorithm is used to build the database for poses recognition, which is provided in the tools of Kinect for windows SDK v2.0 [6]. The interactive system uses this database trained by the algorithm for recognition of discrete gestures in real time and is up to track 6 people at the same time. This system also provides some command voices so that the user can interact with the program such as the possibility of changing different Yoga poses, playing a background music and receiving improvement instructions for a correct position.

II. RELATED WORK

A. *Pose recognition methods by Kinect*

Pose recognition methods focus on detecting human figures in images and video, with the goal that one could decide, for instance, where somebody's elbow appears in a picture. This method does not perceive exactly who is in an image. The algorithm simply estimates where key body joints are. Although, other systems integrate this technology to associate the pose detection with user identifiable information.

Salvatore Source et al. introduced a method to distinguish whether a hand is open or closed base on neural network [7]. They utilize hand division with the RGB frames from Kinect camera and afterward utilize three distinct methodologies as inputs for the Neural classifier. As indicated by their examinations and dialog, their methodology that incorporate hand mask with edge do not get great outcomes and need more autonomous calculations for edge extraction.

Youness Choubik and Abdelhak Mahmoudi proposed to apply machine learning algorithms to classify poses in real time through a characterized terminology, utilizing the skeleton information given by the Kinect sensor [8]. Basically, they extracted features from skeleton data to train machine learning algorithms with different classifiers such as Support Vector Machines, Artificial neural networks, k-nearest neighbors and Bayes classifier. Their application obtained good results, however the poses used for testing were very simple for a convincing study. There was only difference between the positions of the arms.

Yun Han et al. presented an automatic action segmentation and continuous recognition for basic indoor actions based on skeleton data from Kinect [9]. It covers three stages mainly. A state-based modeling of actions that with the inclusion of Finite State Machine (FSM) convert an action recognition problem into a state identification problem. Next stage is an error recovery mechanism since Kinect can lose track of current states. Finally, there is an action segmentation to mark transitions from one action to another that use decision tree. This is a practical solution for constant action monitoring without the need to segment the start and the end of each action to be analyzed.

B. *Yoga frameworks*

YogaST is self-training framework aims at instructing the user/practitioner to accomplish yoga poses accurately and anticipating damage caused by inappropriate stances [10]. They utilized two Kinect 1devices with opposite review headings to acquire the body map. To complete the pose recognition, applied image processing to get contour map and convolved with line masks. With that information, framework extracts the axis body user to compare and analyze the correct posture. The primary trouble is when there is cover of the body parts since division cannot happen accurately.

Eyes-Free Yoga is an exergame that uses depth cameras for visually impaired and low vision practice [11]. This framework utilizes Kinect 1 and the users can interact with a yoga trainer receiving audio instructions for six standing yoga poses. The method is to calculate the different body angles using the Law of Cosines, which was very popular among developers who used this version of Kinect in the beginning for the recognition of gestures. Additionally, through of a normal of 11 rules for each pose, the need of changes in the human body is resolved. It is along these lines, how the input of guidelines for the adjustment of the body is chosen.

Microsoft Kinect device gives an astounding following application advancement interface that empowers human activity investigation like learning Yoga. Despite the fact, there are numerous methodologies and solutions for this research, our interactive system plans to bring a more fast and strong method of recognizing poses.

III. SYSTEM DESCRIPTION

A. *Ada-Boost Algorithm*

AdaBoost is an ensemble method that creates solid classifier from various feeble classifiers. For the most part, it is utilized to get paired arrangement results. Toward the starting, this calculation predicts the original data with the equivalent weight as early learning. After every repetition, the misclassified data is given a high weighting considering the perceptions got while applying the past order rules [12]. In this manner, it keeps on including learners until the point when a strong classifier is come to get high accuracy to the model as depicted in the accompanying advances. In this way, it continues to add learners until a strong classifier is reached in order to get high accuracy to the model as depicted in the next lines.

a) Input the training examples with label {(x1; y1), (x2; y2), …(xm; ym)} where xm is a feature vector in the feature space X and ym is a label from from Y = {-1,+1}.

b) Then initialize all weights D1 where n is the number of elements in the feature vector.

$$D_1(i) = {}^{1}/_{n} \quad (1)$$

c) For t = 1, 2…T. where T is the maximum number of iterations.

- Start training a weak classifier ht using the weights in D1 that means ht ∈ {-1,+1}.
- Calculation of the error related with the weak classifier (the sum of the elements weights that are incorrectly classified):

$$error_t = \sum_{i=0}^{n} D_t(i) h_t(x_i) y_i \quad (2)$$

- Calculation of the weight factor αt for the classifier:

$$\alpha_t = 0.5 \ln\left(\frac{1-error_t}{error_t}\right) \quad (3)$$

- After that, the algorithm updates the weights Dt+1(i) such that Dt+1 is a new distribution and the weights decrease if the element is correctly classified and increase if is incorrectly classified:

$$D_{t+1}(i) = D_t(i)^{-\alpha_t y_t h_t(i)} \quad (4)$$

- Re-normalize the weights:

$$D_{t+1}(i) = \frac{D_{t+1}(i)}{\sum_{i=0}^{n} D_{t+1}(i)} \quad (5)$$

d) Finally, after T iterations, output the final classifier as a linear combination of selected classifiers:

$$H_{(x)} = sign\left[\sum_{i=0}^{n} \alpha_t h_t(x)\right] \quad (6)$$

B. *Training and recognition*

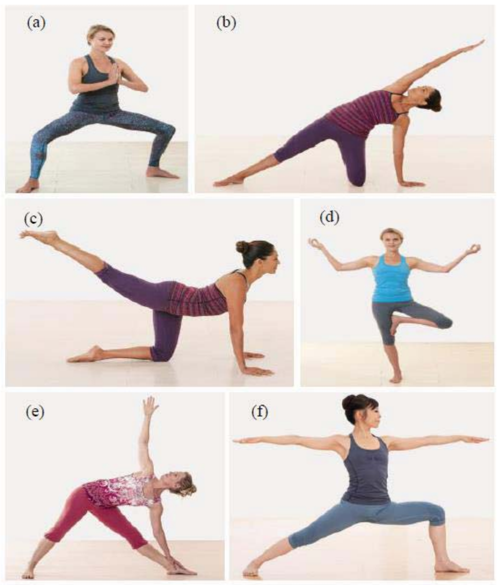


Fig. 1. The six Yoga poses used for the recognition of the system. (a) Dragon flow. (b) Gate pose. (c) Tiger flow. (d) Tree pose. (e) Triangle pose. (f) Warrior flow. For more details, the reader is referred to [13].

In this investigation, 6 Asanas postures based on Hatha Yoga were chosen [13]. Every one of them has an alternate level of trouble and qualities, for example, symmetry, seated not and support on the ground with the hands to analyze the confidence value in the recognition of this framework (See Fig.1).

Kinect for windows SDK v2 has 2 essential tools to build the database of the six yoga poses mentioned above, which are Kinect Studio 2.0 and Visual Gesture Builder (VGB) [14]. The way toward building the database was isolated into 2 phases: Training data and Analysis data. To start, with the assistance of a specialist yoga coach, 5 clips were recorded for every Yoga pose with a framerate of 30 frames/sec.

Through 2 training sessions were acquired a greatest of 10 clips for every Yoga pose in various conditions. From that point forward, the initial 5 clips were incorporated into the training data to be handled by the Ada-boost algorithm. In that way, the initial database was acquired, and we go to the second phase of analysis data. This last data incorporates the other 5 clips and was prepared 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 another one until the point when the last database is made for the all yoga poses. The flowchart for building the database is shown in Figure 2.

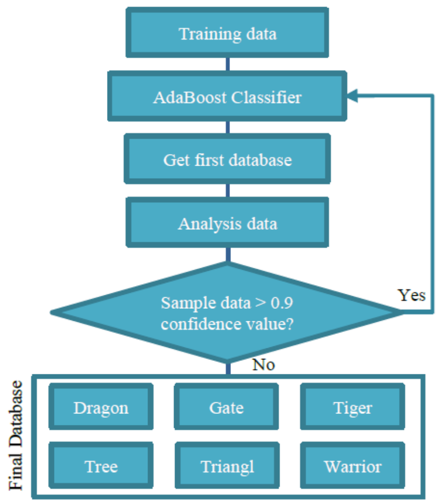


Fig. 2. Flowchart of recognition gestures for database.

Likewise, while making the database, it was produced a log document that shows the 10 weak classifiers by Adaboost algorithm. These weak classifiers are conditioned by parameters, for example, angles, speed and angle velocity. Nevertheless, our study centers around angles between joints since it perceives discrete gestures and do not have to distinguish parameters identified with movements like continuous gestures. VGB trained all data using body frame.

As seen in Table 1, the 3 strong classifiers were obtained after filtering other features and only finding the angles [15]. 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, alternate poses can be assessed by following the classifiers accomplished.

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

Table 1: Top 3 last classifiers for every Yoga Asana acquired by AdaBoost algorithm.

C. *System design*

This interactive system was developed in Visual Studio as WPF [16]. The first step was to initialize Kinect sensor by enabling the 3 streams (colorStream, depthStream and bodyStream) and declaring the parameters used internally in the program. In addition, the speech recognition option is enabled for the following processes. Afterward, the program reads the database that contains the recognition of the 6 Asanas poses obtained from the Adaboost algorithm previously contained in a file called YogaPoses.gbd. The program identifies all the gestures inside this file which are Dragon, Gate, Tiger, Tree, Triangle and Warrior. Then, it is at this moment when the program offers the user to interact with the program through command voices. The options are to show the previous or next Yoga pose, play background music, open and close the main window; and close secondary windows with specific instructions.

For this kind of acknowledgment, it was utilized Microsoft Speech SDK v11.0 [17] because of this model was composed solely for Kinect. Although, other developers can use language models such as Bing Speech, Google Speech or Baidu Speech recognition. In case no voice command is detected, it runs directly to the recognition poses. There is a function called GestureFrameArrived that acquires the actual frame of user and compare with gestures defined in database. This way, it recognizes all poses and updates results. The performance of each user will be detected by the program and it will be recognized which pose is being carried out since it is designed to track up to 6 users at the same time. In the same way, each pose recognition provides 3 important parameters that can be pictured in the primary window which are confidence value, name of pose identified and user detected. In the end, it is analyzed that if the confidence is below the value of 0.75, then the program displays a window with instructions for improvement so that the user has a better performance. These guidelines are beforehand made by the Yoga trainer, so it does not give definite data of the posture made by the client. The flow diagram is shown in Figure 3.

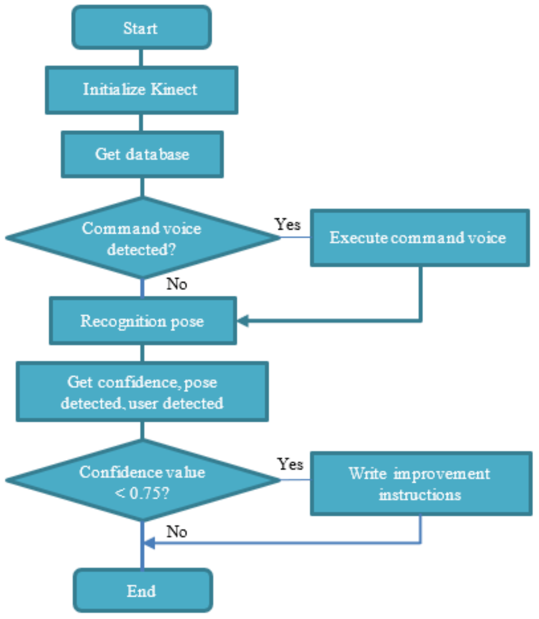
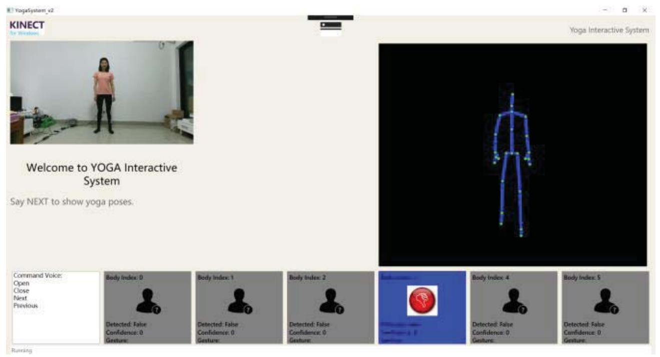


Fig. 3. Flowchart of Yoga interactive system



Welcome to YOGA Interactive System

Say NEXT to show yoga poses.

Command Voice:
Open
Close
Next
Previous

Body Index: 0
Detected: False
Confidence: 0
Gesture:

Body Index: 1
Detected: False
Confidence: 0
Gesture:

Body Index: 2
Detected: False
Confidence: 0
Gesture:

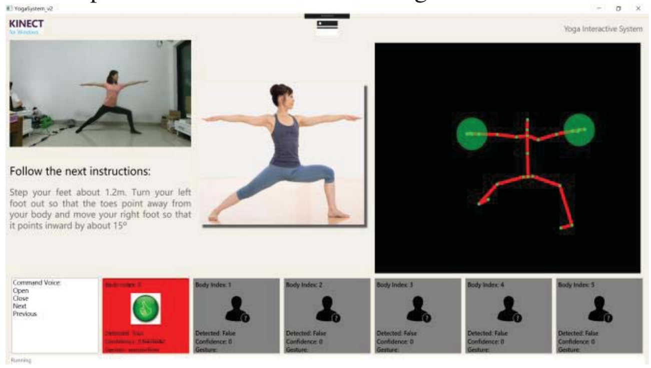
Body Index: 4
Detected: False
Confidence: 0
Gesture:

Body Index: 5
Detected: False
Confidence: 0
Gesture:

Fig. 4. Initial interface of Yoga interactive system.

The main window contains 2 views of cameras that display ColorStream and BodyStream located on the left and right side respectively. In addition, it has a block text that displays the instructions within the program. There are six avatars that represent the maximum number of users that can be detected and a list with the command voices enabled for the interaction with the user at the bottom. Finally, there is a status message that demonstrates if Kinect is running correctly or not connected. (See Figure 4).

When the program starts after say ¨Next¨, the image of yoga pose appears to perform. Afterward, it shows some indications to achieve the pose correctly. If detected, the user avatar will change to a green thumb alongside the name of the represent that has been identified and its certainty level. Else, it will keep on demonstrating the red thumb in the avatar (See Figure 5). Also, if another user appears in front of the camera to be detected, then the program starts tracking this user and shows its own parameters and avatar for recognition.



Follow the next instructions:

Step your feet about 1.2m. Turn your left foot out so that the toes point away from your body and move your right foot so that it points inward by about 15°

Command Voice:
Open
Close
Next
Previous

Body Index: 1
Detected: False
Confidence: 0
Gesture:

Body Index: 2
Detected: False
Confidence: 0
Gesture:

Body Index: 3
Detected: False
Confidence: 0
Gesture:

Body Index: 4
Detected: False
Confidence: 0
Gesture:

Body Index: 5
Detected: False
Confidence: 0
Gesture:

Fig. 5. Yoga interactive system detecting warrior Asana.

IV. EXPERIMENT

In order to evaluate this system, 3 different databases were built in the same way that it was carried out for the training stage of the recognition of poses. In the primary database, just the clips of the yoga mentor of the main session were considered. For the second database, 5 clips of the second session were added and it was built based on the flow diagram presented in Figure 2.

NAME	VALUE	TYPE
Accuracy Level	0.95	FLOAT
Number Weak Classifiers at Runtime	1000	INT
Filter Results	True	BOOL
Auto Find Best Filtering Params	True	BOOL
Weight Of False Positives During Auto Find	0.5	FLOAT
Duplicate And Mirror Data During Training *	False	BOOL
% CPU For Training	95	INT
Use Hands Data	False	BOOL
Ignore Left Arm	False	BOOL
Ignore Right Arm	False	BOOL
Ignore Lower Body	False	BOOL

Table 2: Project settings to build database in VGB by Adaboost algorithm. (*) This parameter is set to true value in triangle and tiger.

The last database was made with the recorded poses of the trainer of both sessions but there was a better precision when tagging the frames with positive results in the Visual Gesture Builder and following the examples prompted by the mentor for each posture. The values set for the parameters in this project are shown in Table 2 and are only used under the algorithm of Adaboost that belongs in the detection of discrete gestures.

When the databases were created, each clip went through a tagging process. In the VGB tooling, there is a control bar with a timeline of all frames to manually tag which of them have a true value for recognition in the current frame selected. All other frames untagged are considered as negative example, but can be tagged as false manually if needed. In Figure 6, the view of the infrared and body are shown, so it will be easy to determine the correct body position in a specific frame [18].

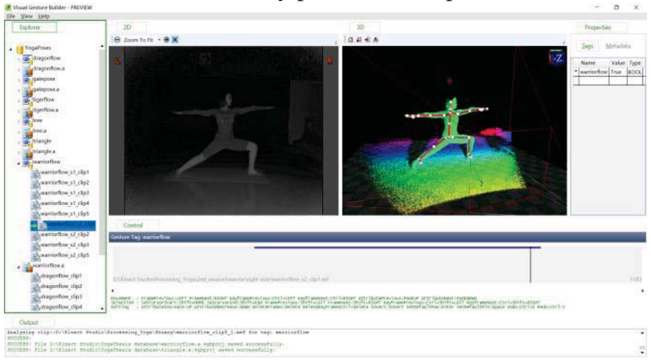
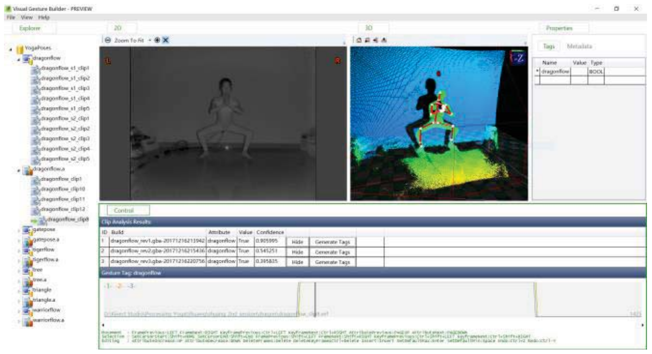


Fig. 6. Tagging warrior pose performance by yoga trainer.

On the other hand, 2 tests were performed by a person who had little knowledge about Yoga and Kinect. The first test was conducted by a subject in front of the Kinect device and only following the instructions of the program. Previously the subject was informed about the objective of this study for a better understanding.

The second test was made by the same user, but this time she received direct instructions from the yoga expert trainer. For each test, 5 to 10 clips were collected which were recorded with the Kinect Studio v2.0 tooling. From them, 5 clips with the best performance were selected and include to a new analysis data. In the control bar, the timeline is displayed with the frames detected as true for the recognition of the poses with their respective confidence value. For instance, Figure 7 shows a clip in which the 3 values of the confidence value were obtained for each database at frame number 1425.



ID	Build	Attribute	Value	Confidence		
1	dragonflow_rev1.gba-20171216213942	dragonflow	True	0.905995	Hide	Generate Tags
2	dragonflow_rev2.gba-20171216215436	dragonflow	True	0.545251	Hide	Generate Tags
3	dragonflow_rev3.gba-20171216220756	dragonflow	True	0.395835	Hide	Generate Tags

Fig. 7. Analysis of dragon pose by amateur subject.

This method is very useful to observe which database provides a better result in general appearance. Therefore, the obtaining of all this data was exported as log file for a detailed statistics analysis. This means that every frame detected as true have a certain confidence value.

V. RESULTS

Figure 8 shows a comparative graph between the timeline of the analysis obtained by VGB and a graph of confidence value vs number of frames generated in the log file of the experiment, since it is desired to represent with greater accuracy the behavior of confidence. In the same way, the results for the other poses were also obtained.

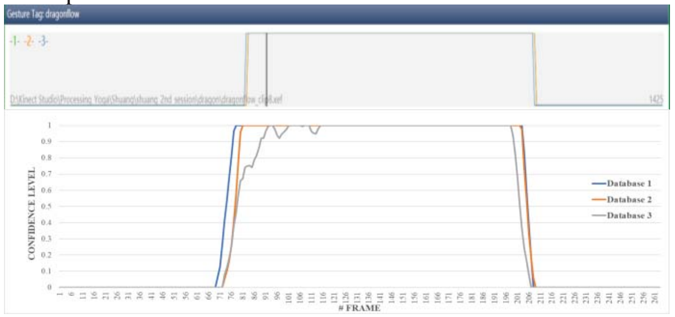


Fig. 8. Chart of confidence value vs frame for dragon pose analysis

After determining all the graphs for poses, Table 3 summarizes the data obtained by analyzing the samples of the amateur subject based on confidence. For a better statistics result, we calculated the number of frames detected by the recognition database as well as mean and the standard deviation of both tests. This mean of confidence represents all values collected from the system in the specific gesture in VGB for each clip recorded. The pooled standard deviation method was used to estimate a single S.D. (σ) to represent all independent samples of each test. The parameter N represents the number of frames detected correctly and that has a value greater than 0.05 of confidence. If it has a lower value, the program assigns that frame as false for the detection.

Table 3: Statistics results based on the confidence of each Yoga poses for the second test.

		N	MEAN (μ)	SD (σ)
DRAGON POSE	DATABASE 1	636	0.9155	0.2120
	DATABASE 2	616	0.7628	0.2090
	DATABASE 3	611	0.6548	0.2200
GATE POSE	DATABASE 1	584	0.7088	0.2390
	DATABASE 2	513	0.4831	0.1810
	DATABASE 3	551	0.5098	0.1890
TIGER POSE	DATABASE 1	413	0.5305	0.2520
	DATABASE 2	223	0.2192	0.1490
	DATABASE 3	282	0.2801	0.1390
TREE POSE	DATABASE 1	127	0.1878	0.1320
	DATABASE 2	198	0.4509	0.1880
	DATABASE 3	129	0.3039	0.1330
TRIANGLE POSE	DATABASE 1	178	0.1532	0.0670
	DATABASE 2	363	0.2008	0.0990
	DATABASE 3	259	0.1150	0.0460
WARRIOR POSE	DATABASE 1	585	0.7060	0.2480
	DATABASE 2	587	0.8805	0.2360
	DATABASE 3	561	0.9211	0.2070

For Dragon pose was obtained a maximum average of 92% for the confidence value. Although database 1 obtains a better number of detected frames, this does not mean that it is the best since in the analysis only 5 clips were taken. On the other hand, database 2 and 3 have a greater number of samples recorded in different situations. Likewise, it is recalled that the database 3 was tagging with greater precision following a pattern in which the hands should be together and not separated. Therefore, database 3 was selected as the best of them.

For Gate pose was obtained a maximum average of 71% for the confidence value. Similar to the previous case, database 1 is discarded since it contains the smallest number of samples. For gate pose, the pattern was to select frames where the right leg and right arm must be parallel in average. Database 3 also was selected for the final.

For Tiger pose, it was mentioned in Table 2 that the parameter will be enabled for duplicate and mirror data during training. This means that it does not matter if the user performs the position with right or left orientation. The program is able to detect it and display the results like any yoga pose. A maximum average of 53% was obtained for the confidence. The pattern used for tagging is that the left leg should be fully stretched and without any flexion in the knee for a right-facing detection. For an orientation to the left, it would be the opposite case. In the end, it was determined to use database 3 because it provided better results.

For tree pose, the maximum average was 45% for the confidence value. For this Yoga pose, there were some problems when trying to keep the right leg on the other thigh. If the user helped their hands to get in this position and maintain balance, the Kinect sensor did not perform a good body tracking of the hand and foot joints since they overlapped and was difficult to detect. The pattern used was the condition previously stated and keep both arms folded perpendicularly with an upward direction. Database 2 was selected.

Triangle is the Yoga pose with less favorable statistics results. A maximum average of 20% was obtained for the confidence level. Like the Tiger pose, the parameters duplicate and mirror data during training was enabled. Therefore, this pose is determined to be evaluated in right or left orientation. However, keeping the arms completely extended forming a triangle with the legs open, did not result in a pose easily detectable for Kinect because the threshold of the angle formed by the legs and arms is variable. Despite this, it was found that the system is capable of detecting even if the confidence value is low. Database 2 was selected for the program.

For Warrior pose, a maximum average of 92% was obtained for the confidence value. Along with dragon pose were the ones that gave the best results. The pattern focused on keeping the head in the right direction and keep the arms fully stretched. Comparing with others, database 3 obtained better efficiency for detection and it was selected.

VI. CONCLUSION

This paper has presented an interactive system that uses Kinect v2 for 6 Yoga poses recognition with command voices to envision the directions and pictures about the stances to be execution. This system can be used as a tool for Yoga teachers or self-learning Yoga who want to improve their position and flexibility and need help to achieve it. Besides, we developed a strong database thought Adaboost algorithm and get higher results based on confidence value and frame detected. For future work, we will expand the interactive system giving more options to the user such as having the choice to save the user's personal data and its performance, comparing the performance of users through scores or making the system more real with the inclusion of virtual reality device that can provides a natural interaction.

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] 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.

[8] Y. Choubik and A. Mahmoudi, "Machine learning for real time poses classification using Kinect skeleton data," *13th International Conference Computer Graphics, Imaging and Visualization*, 2016.

[9] Y. Han, S. Chung and S. Feng, "Automatic action segmentation and continuous recognition for basic indoor actions based on Kinect pose streams," *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2017.

[10] 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.

[11] 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.

[12] 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.

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

[14] 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].

[15] E. Trejo and P. Yuan "Recognition of Yoga poses through an interactive system with Kinect device" 2018 *2nd International Conference on Robotics and Automation Sciences (ICRAS),* 2018.

[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

[18] Kinect for Windows. "Visual Gesture Builder: A Data-Driven Solution to Gesture Detection", Microsoft, 2013