Semantic Description and Recognition of Human Body Poses and Movement Sequences with Gesture Description Language

Tomasz Hachaj[1] and Marek R. Ogiela[2]

[1] Pedagogical University of Krakow, Institute of Computer Science and Computer Methods, 2 Podchorazych Ave, 30-084 Krakow, Poland
tomekhachaj@o2.pl
[2] AGH University of Science and Technology, 30 Mickiewicza Ave, 30-059 Krakow, Poland
mogiela@agh.edu.pl

Abstract. In this article we introduce new approach for human body poses and movement sequences recognition. Our concept is based on syntactic description with so called Gesture Description Language (GDL). The implementation of GDL requires special semantic reasoning module with additional heap-like memory. In the following paragraphs we shortly describes our initial concept. We also present software and hardware architecture that we created to test our solution and very promising early experiments results.

Keywords: Pose recognition, movement sequences recognition, syntactic description, semantic reasoning, natural interface.

1 Introduction

The researches on human - computer interaction have been conducted for many years. The concept of human-device interaction based on human senses, mostly focused on hearing and vision is now known under term Natural Interaction or Natural User Interface (NI). Lately appearing of affordable by a single consumer multimedia sensor resulted in commercialization of NI techniques and intensification of studies on that subject. The vision communication with computer program is mainly based on exposing some predefined body poses and movement sequences. Many methods have been yet proposed for extraction and interpretation of those communicates from video stream. In [1] authors propose a method to quickly and accurately predict 3D positions of body joints from a single depth image using no temporal information. The method is based on body part labeling, extracting depth image features and randomized decision forests. In [2] system for estimating location and orientation of a person's head, from depth data acquired by a low quality device is presented. Approach is based on discriminative random regression forests: ensembles of random trees trained by splitting each node so as to simultaneously reduce the entropy of the class labels distribution and the variance of the head position and orientation. Most of the gesture recognition approaches are based on statistical modeling, such as principal component analysis or hidden Markov models [3]. The concept of modeling the dynamic hand gesture using a finite state machine has been proposed in [4]. In [5] a radial basis function network architecture is developed that learns the correlation of facial feature motion patterns and human expressions. The recognition techniques have many applications not only in games entertainment but also in medicine for example during rehabilitation presses [6]. In this article we introduce new approach for human body poses and movement sequences recognition for NI. Our concept is based on syntactic description with so called Gesture Description Language (GDL) which is LALR(1) grammar. The implementation of GDL requires special semantic reasoning module with additional heap-like memory. In the following paragraphs we shortly describe our initial concept. We also present software and hardware architecture that we created to test our solution and very promising early experiments results.

2 Methods

We have developed the system that allow us initially tests our approach (Figure 1). It is consisted of sensor for data acquisition, image processing library for tracking user and semantic reasoner with GDL language interpreter that is our novel and original contribution. In our solution we utilized the Kinect sensor for data capture and OpenNI Framework software for image processing. We have chosen that software because it has low hardware and software requirements comparing to other solutions with similar capabilities.

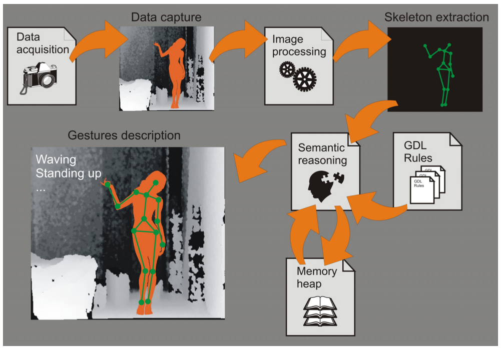

Fig. 1. System architecture. The detailed description is in the text.

2.1 Sensor for Data Acquisition

The Kinect controller was preliminary design to use with the video game and entertainment system. It is consisted of: depth camera, RGB camera and Audio sensors.

2.2 Image Processing (Tracking) Software

The OpenNI framework [7] provides an application programming interface (API) for writing applications utilizing natural interaction. The API enables modules to be registered in the OpenNI framework and used to produce sensory data. PrimeSense NITE Middleware [8] is a module for OpenNI providing gesture and skeleton tracking. Skeleton tracking functionality enables detection and real-time tracking of fifteen key points on human body (see Figure 2). Those key points will be called joints in the rest of the article. After processing of depth sensor data NITE returns joint positions and orientations are given in the real world coordinate system.

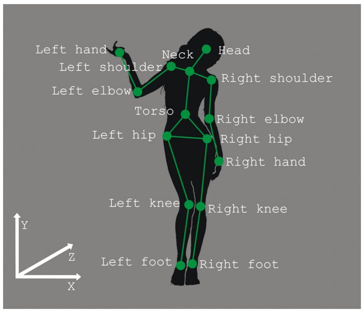

Fig. 2. Skeleton joints and the coordinate system. All body parts are mirrored because user is facing the camera.

2.3 Semantic Reasoner

Semantic reasoner implements our GDL specification. GDL language is used for syntactic description human body poses and movement sequences. The GDL script consists of rules set. Each rule has the logical expression and conclusion. If that expression is satisfied the conclusion is added on the top of memory heap. The conclusion from one rule can be present in logical expression of another rule. Multiple rules can have same conclusion. The determination of truthfulness of all rules is made with forward chaining inferring schema by semantic reasoning module. On each level of the memory heap semantic reasoner keeps information about input data (coordinates of skeleton joints of tracked user) and satisfied rules for given state of memory heap. Each heap level keeps also timestamp informing (how much time passed since last data addition to the top of the heap). Using this time stamp program can easily check how much time has passed from now to the moment when data was added to the chosen heap level simply by summing up all time stamps from to the top to the chosen heap level. GDL gives an access to heap memory by direct access to actual / previous joint coordinate value (the level of the heap must be specified) or by indirect checking if some of possible conclusions was satisfied in the given time period. All possible elements of GDL are presented and described in Table 1. In GDL letter case does not matter. The GDL gives direct access to joint data tracked by OpenNI Framework but our algorithm is not limited to this particular software solution or sensor type. The GDL implementation can be easily adapted to any other image processing framework as long as it generates three dimensional joint-based user tracking data.

Table 1. Elements of GDL

Type	Symbols	Description
Rule definition	**RULE** logicalValue **THEN** conclusion	If logical values equals true the conclusion occurs and it is added on the top of the memory heap.
Relational operators	**<,<=,>,>=,=,!=**	Binary relational operators between numeric values. If the condition is satisfied it returns logical value true, if not false.
Aritmetic operators	**+,-,*,/,%**	Binary arithmetic operators between numeric values. "-" can also be an unary operator.
Logical operators	**&, |**	Logical operators between logical values.
Brackets	()	Bracket can be used for changing the order of operators' execution (arithmetic and logical).
Logical functions	**not()**	Negation of logical value.
Numeric functions	**abs(), sqrt()**	Absolute value and square root of numeric value. If the numeric value in sqrt function parameter is negative it generates programming language exception.
Sequence checking functions	**sequenceexists**("movementSequence")	Returns true if given "movementSequence" exists in the memory heap. Return false in not.
Movement sequence	**"[**conclusion1,!conclusion2,timeRestriction1] …[conclusion3,timeRestriction2]**"**	Sequence of sets of conclusions. Each set of conclusions is in the square brackets. Each conclusion in the set has to be present in the memory heap by the time period specified in field timeRestriction (so the first time period is from current time to current time – timeRestriction1, next period is from current time – time by which all conclusion occurred to current time – time by which all conclusion occurred – timeRestriction2 and so on). If ! precede conclusion that means that given conclusion cannot be present in given time period. If all conditions are satisfied function returns logical value true. If not function returns false.
Existence of the conclusion	conclusion	If conclusion with the given name is present on the top of memory heap it returns logical value true. In not it returns false.

Table 1. (*continued*)

Joint coordinate value	jointName**.x**[heapPosition], jointName**.y**[heapPosition], jointName**.z**[heapPosition]	Returns numeric value of one of joint coordinates (x, y or z). Names of possible joints are presented in Fig. 2. Numeric value heapPosition determinates position of joint to be retrieved from the memory heap (zero means top of the heap). If joint does not exist in the given position of the heap (for example it was not detected by the image processing library) this expression returns 0.
Commentary	// /* */	Single and multiline commentary.

3 Experiment and Results

In order to check the usefulness of proposed semantic description we have created set of scripts that recognize some common behavior that might be present in human - computer interaction. Because of the article space limitation we will only described three of them that represents different capabilities of GDL: detection of movement, recognition of hand clapping and recognition of waving gesture. Our intention was to present not very complex examples to make that approach easier to clarify. It should be remembered that GDL does not limit the number and complexity level or rules in the script. The first example is detection movement along horizontal axis of the tracked user. In that case we assumed that movement means changing position of torso skeleton joint so we have to check if absolute difference value between torsos horizontal coordinates has changed in last two tracking sequences (see Appendix - example 1). The second example is detection of hand clapping along horizontal axis. The proposed implementation is consisted of two frames: with hands separate (the difference between vertical coordinates of hands has to be under given threshold) and hands close to each other - the difference between vertical and horizontal coordinates of the hands are under given threshold (Figure 3, Appendix - example 2). The third rule checks if previous rules have occurred in the specific order in given time constraints. The GDL script example can be easily modified to make recognition independent of coordinate axis by applying Euclidean distance between points.

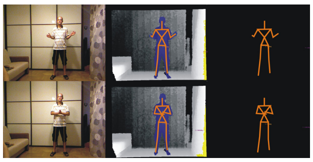

Fig. 3. Two gestures that have been used for description of hand clapping sequence along horizontal axis. In first column user holding his hands separately, the difference between vertical coordinates of hands has to be under given threshold. In second column user holding his hands close to each other - the difference between vertical and horizontal coordinates of the hands are under given threshold. First column – image from RGB camera, second column - image from depth camera with region of interest (user body) and skeleton detected, third column - tracked skeleton.

The third example is gesture of waving with right hand. It has been partitioned to three poses: with right hand over the right elbow, right hand on the left from an elbow and right hand on the right from elbow. The whole sequence should appear in the given time period (Figure 4, Appendix - example 3).

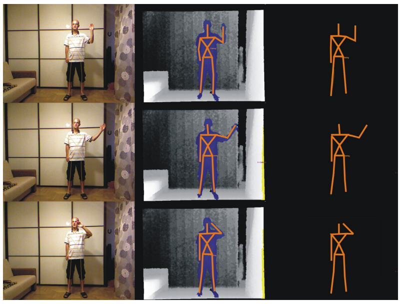

Fig. 4. First column - image from RGB camera, second column - image from depth camera with region of interest (user body) and skeleton detected, third column - tracked skeleton. The detailed description is in the text.

Our experiment has proved that GDL scripts processed by our reasoning framework is capable for real time recognition of the considered gestures. That is because any body position can be expressed by inequalities between selected skeleton joints. The sequenceexists function restricts the time period in which the sequence should appear. The noises generated by tracking software can be compensated by abs function. Our early experiments has also showed that some well know gestures (like those we proposed in this section) are easily reproduced (and recognize by our system) by new users who did not have any previous experiences with tracking software.

4 Discussion

Our preliminary proposition of human body poses and movement sequences recognition system has yet proven to be reliable in recognition of user behavior consisted of few skeleton joints and poses. The main advantages of our approach are simplicity and intuitiveness of GDL scripts. What is more the application of forward chaining inferring schema with memory stack concept is straightforward to any computer programmer and makes development of movement sequences fast and effective. That type of description does not require any training and gathering of huge movement databases. The GDL architect has to decide which skeleton joints can be omitted to simplify the description without affecting resemblance to exact movement recording. Also GDL does not limit the number and complexity level or rules in the script and because semantic analysis of once parsed script is not time demanding our approach can potentially has very large rules databases. The main drawback of the methodology is that complex movement sequences might need many key frames. That problem might be solved by finding solution of reverse problem - automatic generation of GDL from filmed movement sequences (similarly to [9]). With proper computed aided tool that would guide the user in process of removing needles joints and setting tolerance level it might be quite effective. From the other hand it should also be remembered that many multimedia vision systems for natural user interfaces use one camera for capturing the movement of user. That fact strongly limits the field of view of the device leading to limitation in observation of some gestures in particular body positions. In those situations some key points of the body might be invisible and difficult to predict by the software. Because of that in many cases one can omit dependences of skeleton joint towards selected axis because the moving sequence will not be proper reiterated by sensor and tracking software.

5 Conclusion

Our goals for the future will be comparison of our approach to other existing methods and validation its sensitivity on test datasets. We also want to include the functionality that will supply the user with possibility of interacting with virtual object and environment (for example three dimensional visualizations of medical images [10]). That would require adding new procedures to GDL and more complex memory stack architecture. We should also consider adding some languages semantic that would allow user to define parts of code that are used multiply times (for example possibility of variables definition).

Acknowledgments. We kindly acknowledge the support of this study by a Pedagogical University of Krakow Statutory Research Grant.

References

1. Shotton, F., et al.: Real-time human pose recognition in parts from single depth images. In: CVPR 2011 (March 2011)
2. Fanelli, G., Weise, T., Gall, J., Van Gool, L.: Real Time Head Pose Estimation from Consumer Depth Cameras. In: Mester, R., Felsberg, M. (eds.) DAGM 2011. LNCS, vol. 6835, pp. 101–110. Springer, Heidelberg (2011)
3. Mitra, S., Acharya, T.: Gesture recognition: A survey. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 37(3) (2007)
4. Yeasin, M., Chaudhuri, S.: Visual understanding of dynamic hand gestures. Pattern Recognition 33, 1805–1817 (2000)
5. Rosenblum, M., Yacoob, Y., Davis, L.S.: Human expression recognition from motion using a radial basis function network architecture. IEEE Trans. Neural Netw. 7(5), 1121–1138 (1996)
6. Obdrlek, S., Kurillo, G., Han, J., Abresch, T., Bajcsy, R.: Real-Time Human Pose Detection and Tracking for Tele-Rehabilitation in Virtual Reality. Studies in Health Technology and Informatics 173, 320–324 (2012)
7. OpenNI homepage, `http://www.openni.org`

8. Prime Sensor NITE 1.3 Algorithms notes, Version 1.0, PrimeSense Inc. (2010), `http://pr.cs.cornell.edu/humanactivities/data/NITE.pdf`
9. Hong, P., Turk, M., Huang, T.S.: Gesture modeling and recognition using finite state machines. In: Proc. 4th IEEE Int. Conf. Autom. Face Gesture Recogn., Grenoble, France, pp. 410–415 (2000)
10. Hachaj, T., Ogiela, M.R.: Visualization of perfusion abnormalities with GPU-based volume rendering. Computers & Graphics 36(3), 163–169 (2012)

Appendix

Example 1. GDL scripts that detect movement along horizontal axis of the tracked user.

```
RULE abs(torso.x[0] - torso.x[1]) > 10 THEN Moving
```

Example 2. GDL scripts that detect hand clapping along horizontal axis.

```
RULE abs(RightHand.x[0] - LeftHand.x[0]) < 80 & abs(RightHand.y[0] -
LeftHand.y[0]) < 80 THEN HandsTogether
RULE abs(RightHand.x[0] - LeftHand.x[0]) > 80 & abs(RightHand.y[0] -
LeftHand.y[0]) < 80 THEN HandsSeparate
RULE sequenceexists( "[HandsSeparate,0.5] [HandsTogether,0.5] [HandsSep-
arate,0.5]" ) THEN HandClap
```

Example 3. GDL scripts that detect waving with right hand.

```
RULE RightElbow.x[0] > Torso.x[0] & RightHand.x[0] > Torso.x[0] &
RightHand.y[0] > RightElbow.y[0] & abs(RightHand.x[0] - RightElbow.x[0])
< 50 THEN WavingGestureCenter
RULE RightElbow.x[0] > Torso.x[0]& RightHand.x[0] > Torso.x[0] &
RightHand.y[0] > RightElbow.y[0] & RightHand.x[0] - RightElbow.x[0] <= -
50 THEN WavingGestureLeft
RULE RightElbow.x[0] > Torso.x[0] & RightHand.x[0] >  Torso.x[0] &
RightHand.y[0] > RightElbow.y[0] & RightHand.x[0] - RightElbow.x[0] >=
50 THEN WavingGestureRight
RULE sequenceexists ( "[WavingGestureLeft,1] [WavingGestureCenter,1]
[WavingGestureRight,1]" ) THEN WavingRight
RULE sequenceexists ( "[WavingGestureRight,1] [WavingGestureCenter,1]
[WavingGestureLeft,1]" ) THEN WavingLeft Rule WavingRight | WavingLeft
THEN Waving
```