Multimedia Systems (2014) 20:81–99
DOI 10.1007/s00530-013-0332-2

REGULAR PAPER

Rule-based approach to recognizing human body poses and gestures in real time

Tomasz Hachaj · Marek R. Ogiela

Received: 4 February 2013 / Accepted: 9 August 2013 / Published online: 3 September 2013

Abstract In this paper we propose a classifier capable of recognizing human body static poses and body gestures in real time. The method is called the gesture description language (GDL). The proposed methodology is intuitive, easily thought and reusable for any kind of body gestures. The very heart of our approach is an automated reasoning module. It performs forward chaining reasoning (like a classic expert system) with its inference engine every time new portion of data arrives from the feature extraction library. All rules of the knowledge base are organized in GDL scripts having the form of text files that are parsed with a LALR-1 grammar. The main novelty of this paper is a complete description of our GDL script language, its validation on a large dataset (1,600 recorded movement sequences) and the presentation of its possible application. The recognition rate for examined gestures is within the range of 80.5–98.5 %. We have also implemented an application that uses our method: it is a three-dimensional desktop for visualizing 3D medical datasets that is controlled by gestures recognized by the GDL module.

Keywords Gesture recognition · Action recognition · Rule-based approach · Natural user interfaces · Time series analysis · Syntactic approach · Gesture description language

Communicated by B. Prabhakaran.

T. Hachaj (✉)
Institute of Computer Science and Computer Methods, Pedagogical University of Krakow, 2 Podchorazych Ave, 30-084 Kraków, Poland
e-mail: tomekhachaj@o2.pl

M. R. Ogiela
AGH University of Science and Technology, 30 Mickiewicza Ave, 30-059 Kraków, Poland
e-mail: mogiela@agh.edu.pl

1 Introduction

Nearly all contemporary home and mobile computers are equipped with build-in cameras and video capture multimedia devices. Because of that, there is a heavy demand for applications that utilize these sensors. One possible field of application is natural user interfaces (NI). The NI is a concept of human-device interaction based on human senses, mostly focused on hearing and vision. In the case of video data, NI allows user to interact with a computer by giving gesture- and pose-based commands. To recognize and interpret these instructions, proper classification methods have to be applied. The basic approach to gesture recognition is to formulate this problem as a time varying signals analysis. There are many approaches to complete this task. The choice of the optimal method depends on time sequence features we are dealing with.

The common approach to gesture recognition is a statistic/probability-based approach. The strategy for hand posture classification from [1] involves the training of the random forest with a set of hand data. For each posture, authors collected 10,000 frames of data in several orientations and distances from the camera, resulting in a total of 30,000 frames. For each posture, they randomly sample a proportion of frames for training, and the rest are used for testing. The random forests classifier is also used in [2] to map local segment features to their corresponding prediction histograms. Paper [3] describes GestureLab, a tool designed for building domain-specific gesture recognizers, and its integration with Cider, a grammar engine that uses GestureLab recognizers and parses visual languages.

Recognizers created with GestureLab perform probabilistic lexical recognition with disambiguation occurring during the parsing based on contextual syntactic information. In [4], linear discriminant analysis is used for posture classification. Paper [5] presents the experimental results for Gaussian process dynamical model against a database of 66 hand gestures from the Malaysian sign language. Furthermore, the Gaussian process dynamical model is tested against the established hidden Markov model for a comparative evaluation. In [6], observed users' actions are modeled as a set of weighted dynamic systems associated with different model variables. Time-delay embeddings are used in a time series resulting from the evolution of model variables over time to reconstruct phase portraits of appropriate dimensions. Proposed distances are used to compare trajectories within the reconstructed phase portraits. These distances are used to train support vector machine models for action recognition. In paper [7], the authors describe a system for recognizing various human actions from compressed video based on motion history information. The notion of quantifying the motion involved, through the so-called motion flow history (MFH), is introduced. The encoded motion information, readily available in the compressed MPEG stream, is used to construct the coarse motion history image (MHI) and the corresponding MFH. The features extracted from the static MHI and MFH briefly characterize the spatio-temporal and motion vector information of the action. The extracted features are used to train the KNN, neural network, SVM and Bayes classifiers to recognize a set of seven human actions. Paper [8] proposes a novel activity recognition approach in which the authors decompose an activity into multiple interactive stochastic processes, each corresponding to one scale of motion details. For modeling the interactive processes, they present a hierarchical durational-state dynamic Bayesian network.

Gestures might also be recognized using a neural network and fuzzy sets. The system [9] uses fuzzy neural networks to transform the preprocessed data of the detected hand into a fuzzy hand-posture feature model. Based on this model, the developed system determines the actual hand posture by applying fuzzy inference. Finally, the system recognizes the hand gesture of the user from the sequence of detected hand postures. Moreover, computer vision techniques are developed to recognize dynamic hand gestures and make interpretations in the form of commands or actions. In [10], a fuzzy glove provides a linguistic description of the current hand posture given by nine linguistic variables. Each linguistic variable takes values that are fuzzy subsets of a lexical set. From this linguistic description, it must be evaluated if the current posture is one of the hand postures to be recognized.

An alternative approach was proposed in the full body interaction framework (FUBI) which is a framework for recognizing full body gestures and postures in real time from the data of an OpenNI-applicable depth sensor, especially the Microsoft Kinect sensor [11, 12]. FUBI recognizes four categories of posture and gestures: static postures, gestures with linear movement, a combination of postures and linear movement and complex gestures. The fourth type of gestures is recognized using $1 recognizer algorithm which is a geometric template matcher [13]. In [14], to recognize actions, the authors also make use of the fact that each gesture requires a player to move his/her hand between an origin and a destination, along a given trajectory connecting any two given areas within the sensible area. The idea behind the proposed algorithm is to track the extreme point of each hand, while verifying that this point starts its motion from the origin of a given action, completes it in compliance with the destination, and traverses a set of checkpoints set along the chosen trajectory. This scheme can be applied to trajectories of either a linear or a circular shape.

The semantic approach to gesture recognition has a long tradition. Paper [15] presents a structured approach to studying patterns of a multimodal language in the context of a 2D-display control. It describes a systematic analysis of gestures from observable kinematical primitives to their semantics as pertinent to a linguistic structure. The proposed semantic classification of co-verbal gestures distinguishes six categories based on their spatio-temporal deixis. Papers [16] and [17] propose a two-level approach to solve the problem of the real-time vision-based hand gesture classification. The lower level of this approach implements the posture recognition with Haar-like features and the AdaBoost learning algorithm. With this algorithm, real-time performance and high recognition accuracy can be obtained. The higher level implements the linguistic hand gesture recognition using a context-free grammar-based syntactic analysis. Given an input gesture, based on the extracted postures, composite gestures can be parsed and recognized with a set of primitives and production rules. In [18], the authors propose a vision-based system for automatically interpreting a limited set of dynamic hand gestures. This involves extracting the temporal signature of the hand motion from the performed gesture. The concept of motion energy is used to estimate the dominant motion from an image sequence. To achieve the desired result, the concept of modeling the dynamic hand gesture using a finite state machine is utilized. The temporal signature is subsequently analyzed by the finite state machine to automatically interpret the performed gesture.

Nowadays, multimedia NI controllers can be purchased at a relatively low cost and we can clearly see a growing number of scientific publications about, and industry

applications of this technology. The natural user interfaces have many commercial applications—not only in games or entertainment. One of the most promising domains is the realm of medical treatment and rehabilitation. Paper [19] reports using open-source software libraries and image processing techniques which implement a hand tracking and gesture recognition system based on the Kinect device that enables a surgeon to successfully, touchlessly navigate within an image in the intraoperative setting through a personal computer. In [20], a validation of the Microsoft Kinect for the assessment of postural control was tested. These findings suggest that the Microsoft Kinect sensor can validly assess kinematic strategies of postural control. Study [21] assessed the possibility of rehabilitating two young adults with motor impairments using a Kinect-based system in a public school setting. Data showed that the two participants significantly increased their motivation for physical rehabilitation, thus improving exercise performance during the intervention phases.

All approaches utilizing fully aromatized techniques of gestures classification either require very large training and validation sets (consisting of dozens or hundreds of cases) or have to be manually tuned, which might be very unintuitive even for a skilled system user. What is more, it is impossible to add any new gesture to be recognized without additional intensive training of the classifier. These three factors might significantly limit the potential application of these solutions in real-life development to institutions that are able to assemble very large pattern datasets. On the other hand, body gesture interpretation is totally natural to every person and—in our opinion—in many cases there is no need to employ a complex mathematical and statistical approach for the correct recognition. In fact, our research presented in this paper proves that it is possible to unambiguously recognize, in real time (online recognition), a set of static poses and body gestures (even those that have many common parts in trajectories) using forward chaining reasoning schema when sets of gestures are described with an "if-like" set of rules with the ability to detect time sequences.

We focus our efforts on developing an approach called a gesture description language (GDL) which is intuitive, easily considered and reusable for any kind of body gestures. The GDL was preliminarily described in our previous works [22, 23]. However, our earlier publications showed only the basic concept of this approach, without a detailed description of the methodology, without the validation and possible applications.

The main novelty of this paper is a complete description of our GDL language in the form of a LALR-1 grammar, its validation on a large dataset and the presentation of a possible application. We have implemented and tested our approach on a set of 1,600 user recordings and obtained very promising results—the recognition rate ranged from 80.5 to 98.5 %. What should be emphasized is that all errors were caused by the inaccuracy of the user tracking and segmentation algorithm (third party software) or by the low frame rate of recording. We did not exclude any recordings with aberrations of the user posture recognition because methods of that kind have to be verified in a real—not a 'sterile laboratory' environment. We have also implemented an application that uses our method—a three-dimensional desktop for volumetric medical dataset visualization which is controlled by gestures recognized by our approach.

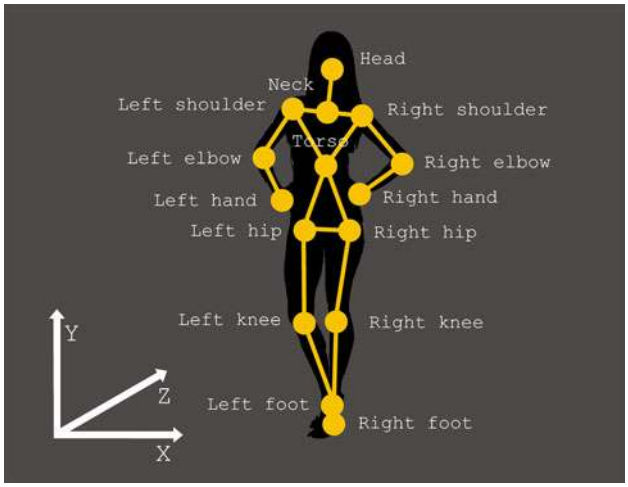

Fig. 1 Body joint positions detected by NITE 1.3 Algorithms

2 Materials and methods

In this section, we have described third party software and libraries that we use to generate the test data for our new method. Later on, we introduce our novel classifier.

2.1 The body position features

In our research, we used the Microsoft Kinect sensor as the device for capturing and tracking human body movements. The body feature points that were utilized in our researches were the so-called skeleton joints. The segmentation of the human body from depth images and the skeleton tracking has been solved and reported in many papers [24–27] before. The joint video/depth rate allocation has also been researched [28]. In our solution, we have utilized the Prime Sensor NITE 1.3 Algorithms which track 15 body joints as presented in Fig. 1.

Body joints are segmented from a depth image produced by a multimedia sensor. Each joint has three coordinates that describe its position in the right-handed coordinate

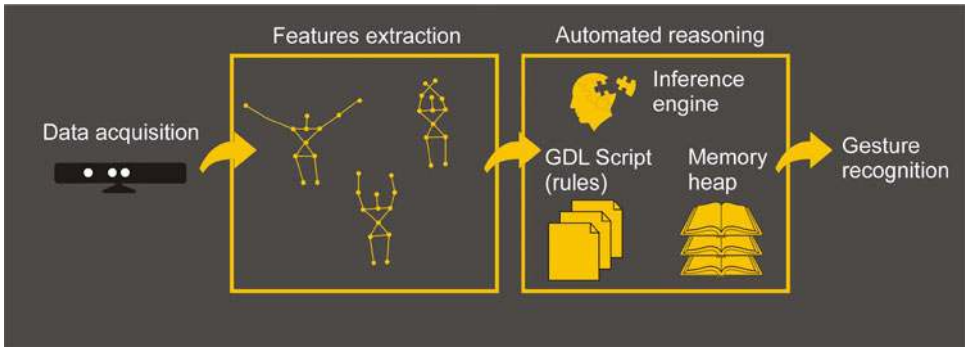

Fig. 2 The schema of the GDL approach. The input data for the recognition algorithm are a stream of body features that arrives continuously from the data acquisition hardware, more details in text

system (note that left and right side in Fig. 1 are mirrored). The origin of this coordinate system is situated in the lens of the sensor's depth camera. After detecting the skeleton joints, NITE tracks those features in real time, and this is crucial for the subsequent gesture recognition.

2.2 Our semantics in a rule-based approach with the GDL

Our approach is based on assumption that (nearly) every person has broad experience in how real-life (natural) gestures are performed by a typical person. Every individual gains this experience subconsciously through years of observation. Our goal was to propose an intuitive way of writing down these observations as a set of rules in a formal way (with some computer language scripts) and to generate a reasoning module that allows these scripts to be interpreted online (at the same speed at which the data arrives from the multimedia sensor). The schema of our approach is presented in Fig. 2. The input data for the recognition algorithm are a stream of body joints that arrives continuously from the data acquisition hardware and third party libraries. Our approach is preliminarily designed to use a set of body joint extracted by the NITE library, but it can be easily adapted to utilize any other joint set tracked by any of the previously mentioned libraries.

The very heart of our method is an automated reasoning module. It performs forward chaining reasoning (similar to that of a classic expert system) with its inference engine any time new portion of data arrives from the feature extraction library. All rules of the knowledge base are organized in GDL scripts which are simply text files that are parsed with a LALR-1 grammar. The input set of body joints and all conclusions obtained from knowledge base are put on the top of the memory heap. In addition, each level of the heap has it own timestamp which allows checking how much time has passed from one data acquisition to another. Because the conclusion of one rule might form the premise of another one, the satisfaction of a conclusion does not necessarily mean recognizing a particular gesture. The interpretation of a conclusion appearance depends on the GDL script code.

The detailed algorithm that leads from the obtained feature points to data recognition is as follow:

1) Parse the input GDL script that consists set of rules, generate a parsed tree.
2) Start the data capture module.
3) Repeat the below instructions until the application is stopped:
   a. If new data (a set of body feature points) arrive from the data capture algorithm, store it on the top of the memory heap with the current timestamp. Each level of the memory heap contains two types of data: a set of feature points and a set of rule names that were satisfied for the current/previously captured feature points and the current/previously satisfied rules. Feature points and names of rules satisfied in previous iterations lie on memory heap layers corresponding to a particular previous iteration. The size of the memory heap in our implementation is 150 layers (5 s of data capture at the frequency of 30 Hz). We did not find a "deeper" heap useful, but because data stored in the heap are not memory consuming, the heap might be implemented as "virtually unlimited in depth".
   b. Check if values of those new points satisfy any one rule in the memory heap whose name is not present in the top level of the memory heap. The GDL script may also consider feature points from previous captures; for example torso.x[0] is the actual x-coordinate of the torso joint while torso.x[2] is the x-coordinate of the same joint but captured two iterations before—two levels below from the top position of the memory heap (see Examples 1, 2). Rule truth might also depend on the current (see Examples 3–5) and previously satisfied rules (see Example 6).
   c. If any rule is satisfied, add its name to the top of the memory heap at same layer at which the last

captured feature points were stored (top level of heap). As each level of the heap has its timestamp, it is possible to check if some rule was satisfied no earlier than a given period of time ago. It is simply done by searching the memory heap starting from the top (and then descending) until the rule name is found in some heap level or you get to before the given time period. Thanks to this mechanism, it is possible to check if the particular sequence of body joint positions (represented by rule names) appeared in the given time period. The sequence of body positions defines the gesture that we want to recognize (see Example 6).

d. If the name of the rule was added to memory heap in step 'c' of the algorithm, go to step 'b'. If no new rule name was added, return to step '3' and wait for new data to arrive.

Another very important part of our approach consists in predefined variables ("Appendix 1", terminal symbols "Body parts") that return the value of body joint coordinates. It is possible to take not only the current position of a joint but also any previous position found in the memory heap. This is done by supplying a predefined variable name with a memory heap index (0 is the top of the heap, 1 is one level below the top, etc.). There is also a set of functions, which returns either numerical or logical values. For example, the following function checks if the Euclidean distance between the current position of the torso (x, y and z coordinate) and the previous position stored one heap level below is greater than 10 mm. If so, the rule is satisfied and the conclusion 'Moving' is added to the memory heap at the level 0. This means that the GDL recognized the movement of the observed user.

Example 1:

```
RULE Distance(torso.xyz[0], torso.xyz[1]) > 10 THEN Moving
```

In the next paragraph we present GDL scripts examples with commentaries that clarify most of possible syntax constructions and the algorithm's flow.

Figure 3 shows how motion can be mapped with a parsed rule from Example 1

The same rule might be rewritten as:

Example 2:

```
RULE sqrt((torso.x[0] - torso.x[1])^2 + (torso.y[0] - torso.y[1])^2 + (torso.z[0] -
torso.z[1])^2) > 10 THEN Moving
```

2.3 GDL script specification

In this section we will formally define the GDL script language that is used to create the knowledge base for the inferring engine. In GDL, the letter case does not matter. The GDL script is a set of rules. Each rule might have an unlimited number of premises that are connected by conjunction or alternative operators. In GDL, premises are called logical rules. A logical rule can take two values: true or false. Apart from logical rules, the GDL script also contains numeric rules (3D numeric rules) which are simply some mathematical operations that return floating-point values (or floating three-dimensional points). A numeric rule might become a logical rule after it is combined with another numeric rule by a relational operator. The brackets in logical and numeric (3D) rules are used to change the order in which instructions are executed.

The more complex rule which detects the so-called "PSI pose" (see Fig. 5, second image from the right in the first row) is presented below:

Example 3:

```
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
& abs(RightShoulder.y[0] - RightElbow.y[0]) < 50 THEN RightHandPsi
RULE LeftElbow.x[0] < Torso.x[0] & LeftHand.x[0] < Torso.x[0]
& LeftHand.y[0] > LeftElbow.y[0] & abs(LeftHand.x[0] - LeftElbow.x[0]) < 50
& abs(LeftShoulder.y[0] - LeftElbow.y[0]) < 50 THEN LeftHandPsi
RULE RightHandPsi & LeftHandPsi THEN Psi
```

The first rule checks if the right elbow and the right hand are situated to the right of the torso, if the right hand is above the right elbow and if the vertical coordinates of the right hand and the right elbow are no more than 50 mm different. The last part of the rule is the premise that checks if the horizontal coordinates of the right shoulder and the right elbow are no more than 50 mm different. The second rule is similar to first one, but it describes the left arm, shoulder and elbow. The last rule checks if both previous rules are satisfied. This is done by checking the logical conjunction of both previous conclusions.

The above approach works fine when the user is standing perpendicular to the camera plane facing the camera, however, it is possible to redefine the set of rules to make the right prediction when the camera is at an angle. To do so, the GDL script introduces the ability to compute

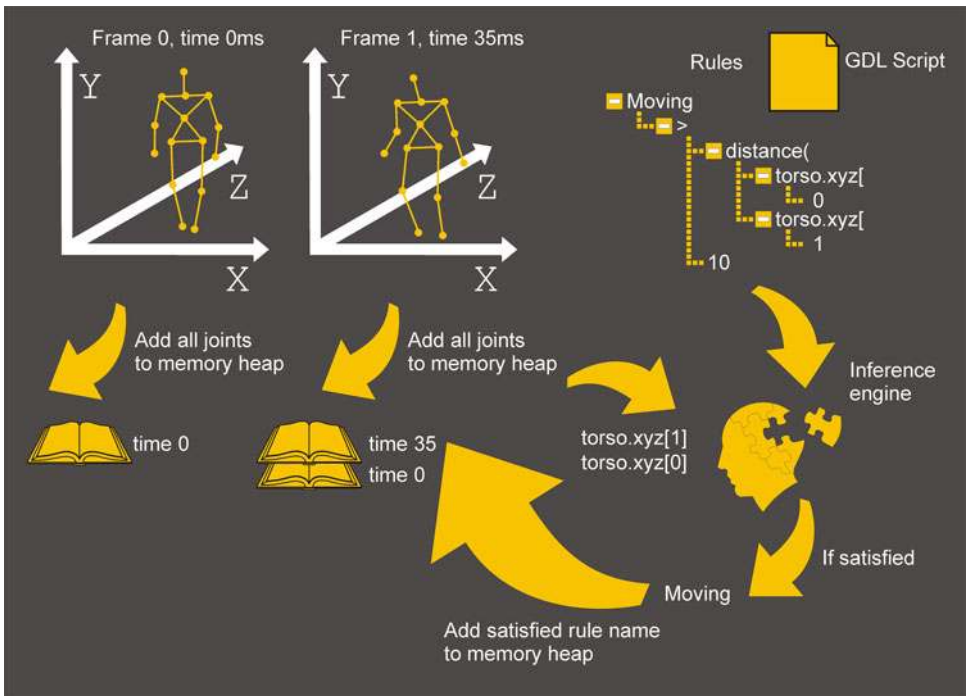

Fig. 3 A simple example of how motion can be mapped with the parsed rule from Example 1. New data (a set of body feature points) arrive from the data capture algorithm and are stored at the top of the memory heap with the current timestamp. The inference engine checks if the rule 'Moving' is satisfied—this requires taking the torso.xyz position from the current and one previous memory heap level. If the Euclidean distance between these positions is greater than 10 mm, then the rule is satisfied and its name is put at the top level layer of the memory heap. If more than one rule is present in the GDL script, the inference engine checks if other rules are now satisfied. Rules that were previously not satisfied might be satisfied now if they are dependent on the presence of 'Moving' at the top of the memory heap (forward chaining reasoning)

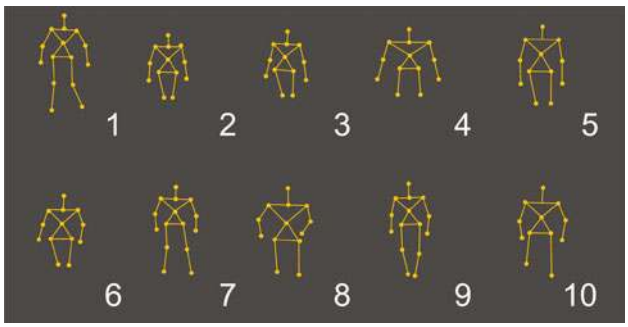

Fig. 4 Different sizes and proportions of detected skeletons build of body joints of the first ten participants in the experiment

the angle between two given vectors in a three-dimensional space using the angle() function (see Example 4). The function takes two parameters that are vectors coordinates and computes the angle between them within the range $[0, 180]$ according to the well-known formula:

$$\angle\overline{a},\overline{b} = \arccos\left(\frac{\overline{a} \circ \overline{b}}{|\overline{a}| \cdot |\overline{b}|}\right)$$

where $\overline{a},\overline{b}$ are vectors, $\circ$ is a dot product and $||$ is a vector's length. Using this approach, we can rewrite the "PSI pose" formula to be independent from the rotation of the camera position around the y-axis (up axis).

Example 4:

```
RULE angle(neck.xyz[0] - RightShoulder.xyz[0],
       RightElbow.xyz[0] - RightShoulder.xyz[0]) > 160
& abs(angle(RightElbow.xyz[0] - RightShoulder.xyz[0],
       RightElbow.xyz[0] - RightHand.xyz[0]) -90) < 20
& RightHand.y[0] - RightElbow.y[0] > 0 THEN RightHandPsi
RULE angle(neck.xyz[0] - LeftShoulder.xyz[0],
       LeftElbow.xyz[0] - LeftShoulder.xyz[0]) > 160
& abs(angle(LeftElbow.xyz[0] - LeftShoulder.xyz[0],
       LeftElbow.xyz[0] - LeftHand.xyz[0]) -90) < 20
& LeftHand.y[0] - LeftElbow.y[0] > 0 THEN LeftHandPsi
RULE RightHandPsi & LeftHandPsi THEN PsiInsensitive
```

The first rule checks if the angle between the vector defined by the neck and the right shoulder and the vector defined by the right elbow and the right shoulder is greater than 160° (both vectors are nearly parallel). The second rule checks if the vector defined by the right elbow and the right shoulder and the vector defined by the right elbow and the right hand are perpendicular. This rule also checks if the right hand is above the elbow. The second rule is similar to first but it applies to the left hand. The last rule is true if both previous rules are satisfied. If the conclusion *PsiInsensitive* is true, this means that the gesture was classified.

The remaining GDL scripts introduced in this work are also insensitive to the *y*-axis rotation. If the camera up-vector rotates around the *x*- or (and) *y*-axis and the angles of rotation are known, it is easy to use the linear transformation to recalculate the coordinates of observed points to the Cartesian frame, in which the camera up-vector is perpendicular to the ground plane. For this reason, we did not consider any rotation other than *y*-axis rotation in the proposed descriptions.

The Example 5 resolves Example's 4 angle issue between the subject position and the camera. Instead of checking if both left and right hands are above elbows GDL script examines if distance between right hand and head is smaller than distance between right hand and right hip (similarly for left hand and left hip). If it is true we know that from two possible hands orientation that satisfies previous conditions hands are above (not below) elbows.

Example 5:

```
RULE angle(neck.xyz[0] - RightShoulder.xyz[0],
RightElbow.xyz[0] - RightShoulder.xyz[0]) > 160
& abs(angle(RightElbow.xyz[0] - RightShoulder.xyz[0],
RightElbow.xyz[0] - RightHand.xyz[0]) -90) < 20
& distance(RightHand.xyz[0], Head.xyz[0]) < distance(RightHand.xyz[0], RightHip.xyz[0])
THEN RightHandPsi
RULE angle(neck.xyz[0] - LeftShoulder.xyz[0],
LeftElbow.xyz[0] - LeftShoulder.xyz[0]) > 160
& abs(angle(LeftElbow.xyz[0] - LeftShoulder.xyz[0],
LeftElbow.xyz[0] - LeftHand.xyz[0]) -90) < 20
& distance(LeftHand.xyz[0], Head.xyz[0]) < distance(LeftHand.xyz[0], LeftHip.xyz[0])
THEN LeftHandPsi
RULE RightHandPsi & LeftHandPsi THEN PsiInsensitive2
```

The last very important ability of GDL scripts to check the presence of particular sequences of body joints that appeared in a constrained time range. A gesture is defined in GDL as a series of static poses (so-called key frames) appearing one after another within given time constraints. The example below checks if the tracked user is clapping his/her hands:

Example 6:

```
RULE distance(RightHand.xyz[0], LeftHand.xyz[0]) < 100 THEN HandsTogether
RULE distance(RightHand.xyz[0], LeftHand.xyz[0]) >= 100 THEN HandsSeparate
RULE sequenceexists("[HandsSeparate,0.5][HandsTogether,0.5][HandsSeparate,0.5]") THEN Clapping
```

The first rule checks if the distance between hands is shorter then 10 cm, the second rule checks if it is greater than 10 cm. The last rule checks if the following sequence is present in the memory heap: the *HandsSeparate* pose has to be found in the heap no earlier than half a second ago, then the time between the *HandsSeparate* and *HandsTogether* (*HandsTogether* had to appear before *HandsSeparate*) poses cannot exceed half a second and the time between *HandsTogether* and the second appearance of *HandsSeparate* cannot exceed half a second. It can be seen that the sequence of gestures is described from the one that should appear most recently to the one that should have happened at the beginning of the sequence. That is because the sequence describes the order of conclusions on the memory heap starting from the top and going to lower layers. The sequenceexists function returns the true logical value if all conditions are satisfied and the false value if any of the conditions in the time sequence is not satisfied. If the conclusion *Clapping* appears in the memory heap, this means that the gesture was identified.

The complete specification of GDL scripts is presented in "Appendix 1". We do not describe all possible contractions, operators and functions here because their names and roles are similar to typical instructions from other programming languages (like JAVA, C++ or C#).

3 The experiment and results

In this section we will describe the experiment we have carried out to validate the performance of our rule-based classifier.

3.1 Experiment setup

To validate our approach we collected a test set. The set consisted of a recording of ten people (eight men and two women) who made four types of gestures (clapping, raising both hands up simultaneously—"Not me", waving with both hands over the head—"Jester" and waving with the right hand) and another five men and five women who made another

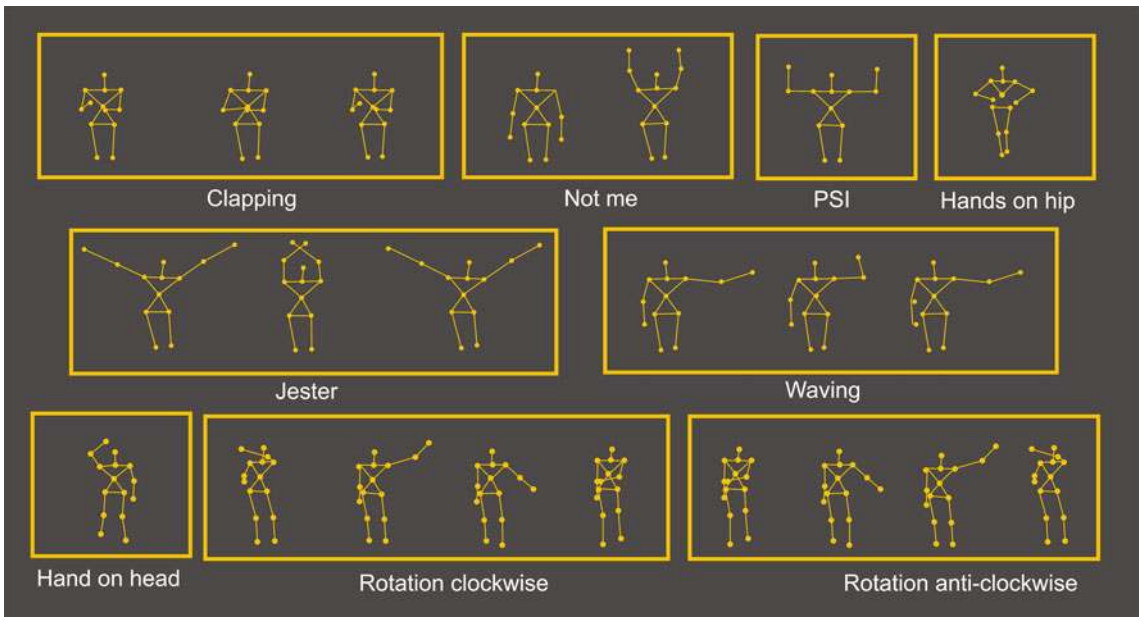

Fig. 5 'Key frames' for the GDL script of gestures. The *right side* of the body in the image represents the *left side* of the recorded participant's body

four types of gestures (touching the head with the left hand—"Head", touching hips with both hands—"Hips", rotating the right hand clockwise—"Rot" and rotating the right hand anti-clockwise—"Rot anti"). Each person made each gesture 20 times, which means the whole database contains 1,600 recordings. All experiment participants where adults of different ages, postures (body proportions) and fitness levels (the test set consisted of both active athletes—for example a karate trainer—and persons who declare they had not done any physical exercise from a long time).

The random error of the depth measurement by our recording hardware (Kinect) increases along with the increasing distance from the sensor, and ranges from a few millimeters up to about 4 cm at the maximum range of the sensor [29, 30]. This error affects the NITE joint segmentation algorithm, causing the constant body point distance to fluctuate over time. Because of the presence of this random error, we did not consider adding other noise to the recorded dataset. Because our approach is time-dependent, any delay in recognition by a time exceeding rule constraints or any errors in motion transitions will disturb the final recognition. To check how much these time constraints affect the final results, the recording was made at two speeds: a low frame rate acquisition speed (7–13 fps for 6 persons in the test set) and a high frame rate acquisition speed (19–21 fps for 14 persons in the test set). There was no single, particular distance between the persons and the video sensor; the only requirement was that the body parts above knees had to be captured by the device. Individuals taking part in the experiment declared that they had no previous experience with this kind of technology. Figure 4 presents one sample frame of the first ten participants of the experiment. The figure shows different sizes and proportions of detected skeletons build of body joints. The difference in size is the result of no restriction policy concerning the distance between the sensor and the individual that was recorded. In addition, in seven cases the NITE algorithm failed to detect the users' feet.

The participants were asked to make the following gestures (see Fig. 5):

- Clapping;
- "Not me" gesture: raising both hands simultaneously above the head;
- "Jester" gesture: waving with both hands above the head. Both hands have to be crossed when they arrive above the head (see Fig. 6, frames 10, 11, 12). The complete example "Jester" sequence recorded for one experiment participant is shown in Fig. 6.
- Waving with the right hand;
- "Head" gesture: touching the head with the left hand;
- "Hips": touching hips with both hands simultaneously;
- "Rotation anticlockwise" ("Rot-anti"): rotating the right hand in the anti-clockwise direction;
- "Rotation clockwise" ("Rot"): rotating the right hand in the clockwise direction.

Key frames of these gestures are presented in Fig. 5. These particular gestures have been chosen because:

- Some movements have similar joint trajectories (for example the middle part of the Jester could be recognized as Clapping, also the Waving could be a part of Jester—see "Appendix 2" for explanations)

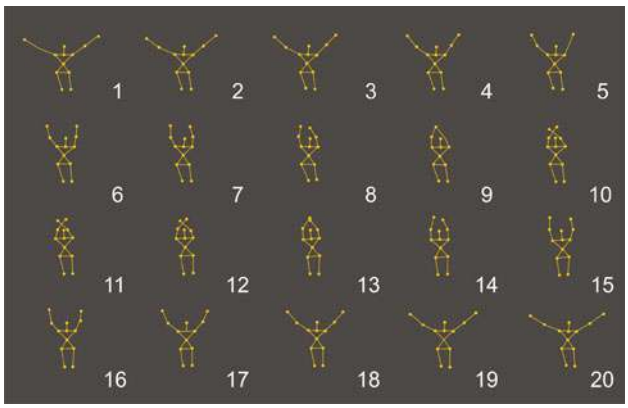

Fig. 6 An example "Jester" sequence recorded for one of the experiment participants

which makes it more difficult to recognize them unambiguously.

– All gestures are based on arm movements, which make the gestures easier to make for test participants, as a result of which they may be more relaxed and make the gestures naturally. We have observed that this relaxation makes the people start performing these gestures in harmony with their body language and the details of recordings differ much between participants.

The gestures from the validation set may be divided into two groups: those that can be described with one key frame (static gestures—Hands on hips and Hand on head) and those that need more than one key frame (Clapping, Not me, Jester, Waving, Rotation clockwise, Rotation anti-clockwise). The architecture of the GDL script for the first group of gestures is very simple—they require only one GDL rule (the key frame) to be recognized. The gestures subject to the second rule require a number of key frames (each has to be represented by a GDL rule) and a number of rules that define the sequence of key frames over time. Because the construction methodology of the GDL script is quite similar for the considered gestures, we will discuss only three selected ones (Not me, Jester and Rotation clockwise) in detail. The remaining GDL scripts used for recognizing the validation set will be presented with a short commentary in "Appendix 2".

The GDL script we use for recognizing the "Not me" gesture (see Fig. 5, first row, second from the left) is presented below:

```
//"Not me" gesture.
RULE RightHand.y[0] < RightElbow.y[0] & RightElbow.y[0] < Torso.y[0]
& LeftHand.y[0] < LeftElbow.y[0] & LeftElbow.y[0] < Torso.y[0] THEN handsDown
RULE RightHand.y[0] > RightElbow.y[0] & RightHand.y[0] > RightShoulder.y[0]
& LeftHand.y[0] > LeftElbow.y[0] & LeftHand.y[0] > LeftShoulder.y[0] THEN handsUp
RULE sequenceexists("[handsUp,0.5][handsDown,0.5]") THEN NotMe
```

The GDL script description of this gesture is very straightforward. It consists of two rules that describe the key frames (with conclusions *handsDown* and *handsUp*) and one rule that checks if the proper sequence of those frames is present in the heap (with the conclusion *NotMe*)—at first the hands should be down along the body, than the hands are raised above the body. If the *NotMe* conclusion appears in the memory heap, this means that the gesture was recognized.

The GDL script we use for recognizing the "Jester" gesture (see Fig. 5, middle row, first from the left) is presented below:

```
//"Jester" gesture.
RULE distance(RightHand.xyz[0], RightShoulder.xyz[0]) >
       distance(RightHand.xyz[0], LeftShoulder.xyz[0])
       & distance(LeftHand.xyz[0], LeftShoulder.xyz[0])
       > distance(LeftHand.xyz[0], RightShoulder.xyz[0]) THEN HandsReverse
RULE angle(RightHand.xyz[0] - RightElbow.xyz[0],
       RightShoulder.xyz[0] - RightElbow.xyz[0])  > 140
       & RightHand.y[0] > Head.y[0] & abs(RightHand.y[0] - LeftHand.y[0]) < 100
       & not(HandsReverse) THEN jester1
RULE angle(RightHand.xyz[0] - RightElbow.xyz[0],
       RightShoulder.xyz[0] - RightElbow.xyz[0]) > 90
       & RightHand.y[0] > Head.y[0]
       & abs(RightHand.y[0] - LeftHand.y[0]) < 100 &  HandsReverse THEN jester2
RULE sequenceexists("[jester2,1][jester1,1][jester2,1]") THEN jester11
RULE sequenceexists("[jester1,1][jester2,1][jester1,1]") THEN jester22
RULE jester11 | jester22 THEN Jester
```

The "Jester" gesture is represented by three key frames and two sequences. The first key frame is described by the rule that checks if hands are crossed

over the head (this is the middle frame from Fig. 5 in the jester sequence)—its conclusion is *HandsReverse*. The second key frame is represented by a rule which is satisfied if both hands are at the same vertical height spread above the head to the sides (*jester1*). The last key frame checks if hands are above head, arms are bent at elbows and *HandsReverse* is satisfied. The first sequence in the rule with the conclusion *jester11* checks if conclusions of rules *jester2*, *jester1* and *jester2* are present in the memory heap in the above order within a given time limit. This means that the observed user first spread their hands, that crossed them above their head and then spread them again. The second rule with sequenceexists functions (with the conclusion *jester22*) verifies whether the observed user first crossed their hands, then spread them, and then crossed them again above their head. The last rule recognizes the Jester gesture if any of previous sequences was detected.

The recognition of the circular trajectory, for example rotating the right hand clockwise, can be obtained by separating the gesture into at least four key frames (see Fig. 5, bottom row, second from the left).

```
//Rotation of right hand clockwise
RULE angle(Neck.xyz[1] - RightShoulder.xyz[1], RightShoulder.xyz[1] - RightElbow.xyz[1])
       - angle(Neck.xyz[0] - RightShoulder.xyz[0],
       RightShoulder.xyz[0] - RightElbow.xyz[0]) > 0
       & angle(Neck.xyz[2] - RightShoulder.xyz[2],
       RightShoulder.xyz[2] - RightElbow.xyz[2])
       - angle(Neck.xyz[1] - RightShoulder.xyz[1],
       RightShoulder.xyz[1] - RightElbow.xyz[1]) > 0 THEN AnglGoesRight
RULE angle(Neck.xyz[1] - RightShoulder.xyz[1], RightShoulder.xyz[1] - RightElbow.xyz[1])
       - angle(Neck.xyz[0] - RightShoulder.xyz[0],
       RightShoulder.xyz[0] - RightElbow.xyz[0]) < 0
       & angle(Neck.xyz[2] - RightShoulder.xyz[2],
       RightShoulder.xyz[2] - RightElbow.xyz[2])
       - angle(Neck.xyz[1] - RightShoulder.xyz[1],
       RightShoulder.xyz[1] - RightElbow.xyz[1]) < 0 THEN AnglGoesLeft

RULE RightHand.y[0] - RightHand.y[1] > 10
       & RightHand.y[1] - RightHand.y[2] > 10 THEN HandGoesUp
RULE RightHand.y[0] - RightHand.y[1] < 10
       & RightHand.y[1] - RightHand.y[2] < 10 THEN HandGoesDown

RULE HandGoesUp & AnglGoesRight & RightHand.y[0] > RightShoulder.y[0] THEN RotC1
RULE HandGoesDown & AnglGoesRight & sequenceexists("[RotC1,0.5]")
       & RightHand.y[0] > RightShoulder.y[0] THEN RotC2
RULE HandGoesDown & AnglGoesLeft & sequenceexists("[RotC2,0.5]")
       & RightHand.y[0] < RightShoulder.y[0] THEN RotC3
RULE HandGoesUp & AnglGoesLeft & sequenceexists("[RotC3,0.5]")
       & RightHand.y[0] < RightShoulder.y[0] THEN RotationClockwise
RULE sequenceexists("[RotationClockwise,1]") & not(sequenceexists("[HandsReverse,4]"))
       & not(sequenceexists("[handsUp,4]")) THEN RotationClockwise
```

The above script introduces four 'helper' rules: the first which checks whether the shoulder muscle is expanding (with the conclusion *AnglGoesRight*), the second which checks if the shoulder muscle is contracting (with the conclusion *AnglGoesLeft*), the third that checks whether the hand is moving up (*HandGoesUp*) and the last that checks if the hand is going down (*HandGoesDown*). The clockwise rotation is the sequence of the following conditions (compare with Fig. 5, bottom row, second from left):

- Shoulder muscle is expanding while the hand is going up over the shoulder;
- Then the shoulder muscle is expanding while the hand is moving down;
- Then the shoulder muscle is contracting while the hand is going down;
- And finally the shoulder muscle is contracting while the hand is moving up.

After the last rule, if the previous ones were satisfied, the hand might be in a similar position as in the beginning after making a full clockwise circle (the condition *RotationClockwise* is added to the memory heap).

3.2 Validation

The results of the validation process on the dataset described in the previous chapter are presented in Tables 1, 2, 3, 4, 5 and 6. The recognition was made with GDL scripts from the previous section and from "Appendix 2". We did not exclude any inaccuracy caused by the tracking software from video sequences. The common problem was that the tracking of body joints was lost when one body part

was covered by another (e.g., while the hands were crossed during the "Jester") or that the distance between joints was measured inaccurately when one body part was touching another (for example during hand clapping). What is more, even if the tracked target is not moving, the position of the joints may change in time (the inaccuracy of the measuring hardware). Of course, all these phenomena are unavoidable while working with any hardware/software architecture and they should be filtered by recognition system.

Each row in the table refers to a particular gesture. Each column gives the classification result. The number in the field shows how many recordings were recognized as the given class. As we can easily see, if a value shows up in the diagonal of the table, this means that the recognition was correct. If it is not in the diagonal field, either the gesture was recognized as another class or was not recognized at all. Tables 1, 3 and 5 sum up the total number of recognitions. Tables 2, 4 and 6 present the average number of recognitions (as a percentage of classifications to a given class) plus/minus the standard deviation. Tables 1 and 2 are for data captured at the speed of between 7 and 13 fps, Tables 3 and 4 for data captured at between 19 and 21 fps. Results in Tables 5 and 6 are calculated for the union of these two sets.

The sums of values in table rows might be greater than 100 % (or the overall count of recordings). That is because a single gesture in the recording might be recognized as multiple gestures which match a GDL description. For example, in three instances the "Not me" gestures were simultaneously recognized as the "Not me" and "Clapping" gestures (see Table 1). GDL has the ability to detect and classify many techniques described by the GDL script rule in one recording. If a technique was correctly classified but an additional behavior—actually not present—was also classified, this case was called an excessive misclassification. All of them are marked by brackets in Tables 1, 3 and 5.

Figure 7 is a plot that compares results from Tables 2, 4 and 6.

4 Discussion

The results from the previous chapter allow us to discuss the efficacy of the classifier and propose a practical application for it.

4.1 Results and discussion

As can be clearly seen, the data acquisition speed strongly impacts the classification efficacy. For the considered gestures, the recognition rate in the high-speed video was between 1 % and nearly 20 % higher than in the low-speed video. In addition, the higher standard deviation in low-speed results (higher than in high-speed samples) shows that the classifier is more stable when frames are captured at the speed of 20 fps. The only remarkable exception is the clockwise rotation of the right hand, where the recognition rate in slow-speed samples was 92.5 ± 5.3 %, while that in high-speed samples was 77.5 ± 2.3 %. This result was caused by two participants in the low-speed sample who were performing their gestures very precisely and carefully. What is interesting is that they had more difficulties with the anti-clockwise rotation than participants from the high-speed dataset. What is very important is that—according to our observations—all errors were caused by inaccuracies of the tracking software. Even though all participants were making the gestures as we expected them to, the key frames didnot appear in the

Table 1 Validating the proposed approach (the recording speed was between 7 and 13 fps)

Actual condition	Recognition result								
	Clapping	"Not me"	"Jester"	Waving	Head	Hips	Rotation anticlockwise	Rotation clockwise	No recognition
Clapping	63	0	0	0	2	0	0	0	15
"Not me"	4 (3)	67	0	3	5	0	0	0	4
"Jester"	2	0	73	0	0	0	0	0	5
Waving	0	0	0	66	0	0	0	0	14
Head	0	0	0	0	40	0	0	0	0
Hips	0	0	0	0	0	31	0	0	9
Rotation anticlockwise	0	1	0	0	0	0	29	0	10
Rotation clockwise	0	0	0	2 (2)	0	0	0	37	3

Values in table fields are the total numbers of cases from a dataset that were classified to a given class. The recording speed was between 7 and 13 fps

Table 2 Validating the proposed approach (the recording speed was between 7 and 13 fps)

Actual condition	Recognition result								
	Clapping	"Not me"	"Jester"	Waving	Head	Hips	Rot-anti	Rot	No recognition
Clapping	78.8 ± 5.4 %	0	0	0	2.5 %	0	0	0	18.8 ± 5.5 %
"Not me"	5.0 %	83.8 ± 3.3 %	0	3.8 ± 0.9 %	6.3 ± 2.7 %	0	0	0	5.0 %
"Jester"	2.5 %	0	91.3 ± 1.2 %	0	0	0	0	0	6.3 ± 1.4 %
Waving	0	0	0	82.5 ± 8.8 %	0	0	0	0	17.5 %
Head	0	0	0	0	100.0 %	0	0	0	0
Hips	0	0	0	0	0.00 %	77.5 ± 1.8 %	0	0	22.5 ± 1.8 %
Rotation anticlockwise	0	2.5 %	0	0	0	0	72.5 ± 5.3 %	0	25.0 ± 3.5 %
Rotation clockwise	0	0	0	5.0 %	0	0	0	92.5 ± 5.3 %	7.5 %

Values in the table represent the percentage of a given dataset that was classified to given class (accuracy) ±SD. The recording speed was between 7 and 13 fps

memory stack. Situations like these are, of course, unavoidable. However, the overall results of our classifier are satisfactory (Tables 5, 6). The recognition rate for all the gestures ranged from 80.5 to 98.5 %. This allows multimedia applications that use our methodology to support the user in a convenient way. We also suppose that if the user gets familiar with this kind of NI and learns how to use it, the recognition error for a skilled operator may drop even below the measured values.

Our approach has some limitations. It is difficult to use the GDL script to properly describe a gesture that requires some mathematical formulas (like the collinearity) to be checked. That is a direct result of the nature of the GDL script description that takes into account some key frames of gestures but not the whole trajectory. On the other hand, that is also one of the biggest advantages of our approach, because it makes the recognition highly resistant to body joint tracking noises which are a very important problem. Our approach also enables the algorithm to perform at very low new-data arrival frequencies (13 fps and less) without losing much recognition accuracy. GDL also eliminates the problem of body proportions between users. Of course, if the key point of the gesture is not registered by the tracking algorithm, the gesture cannot be classified. This situation is obvious in the case of recognizing clapping: at low registration frequencies, the recognition ratio falls to 78.8 ± 5.4 %, while at 20 fps it amounts to 97.5 ± 0.7 %.

The number of excessive misclassification cases is 14 for 1,600 recordings in total (0.88 %). Because the number of these errors is below 1 %, we omitted this phenomenon from data analysis. What is more, the ability to recognize multiple gestures in one recording is a major advantage of the GDL script, but in this particular experiment we did not want to recognize one gesture as a part of another. What is important is that we have proven with our research that with properly constructed GDL script rules, the excessive misclassification error will not disturb the recognition results significantly.

When there are multiple similar gestures the effectiveness of recognition is based on the way the rule is written. Overlapping rules introduce a new challenge and handling them could result in the recognition slowing down to the rate of the longest gesture. A hierarchical structure for rules could be explored to resolve the overlapping gesture issue, instead of a list of rules structure adopted presently. As far as the actor being parallel to the camera axis and perpendicular to the camera plane, the problem introduced is not only the realignment of the skeleton to a standard translation and rotation invariant co-ordinate system but also the recognition of the skeleton itself, due to the large amount of self-occlusion the skeleton returned would be highly erroneous.

Table 3 Validating the proposed approach (the recording speed was between 19 and 21 fps)

Actual condition	Recognition result								
	Clapping	"Not me"	"Jester"	Waving	Head	Hips	Rotation anticlockwise	Rotation clockwise	No recognition
Clapping	117	0	0	0	0	0	0	0	3
"Not me"	0	120	0	0	0	0	0	0	0
"Jester"	5	0	111	0	0	0	0	0	4
Waving	0	0	0	120	0	0	5 (5)	1 (1)	0
Head	0	0	0	0	157	0	0	0	3
Hips	0	0	0	0	0	160	0	0	0
Rotation anticlockwise	0	0	0	4 (2)	0	0	146	0	12
Rotation clockwise	1	0	0	2 (1)	0	0	0	124	34

Values in table fields are the total numbers of cases from the dataset that were classified to a given class. The recording speed was between 19 and 21 fps

Table 4 Validating the proposed approach (the recording speed was between 19 and 21 fps)

Actual condition	Recognition result								
	Clapping	"Not me"	"Jester"	Waving	Head	Hips	Rot-anti	Rot	No recognition
Clapping	97.5 ± 0.7 %	0	0	0	0	0	0	0	2.5 ± 0.6 %
"Not me"	0	100.0 %	0	0	0	0	0	0	0
"Jester"	4.2 ± 0.7 %	0	92.5 ± 1.9 %	0	0	0	0	0	3.3 ± 0.7 %
Waving	0	0	0	100.0 %	0	0	4.2.0 ± 2.7 %	0.8 ± %	0
Head	0	0	0	0	98.1 ± 0.7 %	0	0	0	1.9 ± %
Hips	0	0	0	0	0	100.0 %	0	0	0
Rotation anticlockwise	0	0	0	2.5 %	0	0	91.3 ± 2.1 %	0	7.5 ± 2.1 %
Rotation clockwise	0.6 %	0	0	1.3 %	0	0	0	77.5 ± 2.3 %	21.3 ± 1.9 %

Values in the table represent the percentage of the given dataset that was classified to a given class (accuracy) ±SD. The recording speed was between 19 and 21 fps

Table 5 Validating the proposed approach (the data comes from both low and high-speed datasets)

Actual condition	Clapping	"Not me"	"Jester"	Waving	Head	Hips	Rotation anticlockwise	Rotation clockwise	No recognition
Clapping	180	0	0	0	2	0	0	0	18
"Not me"	4 (3)	187	0	3	5	0	0	0	4
"Jester"	7	0	184	0	0	0	0	0	9
Waving	0	0	0	186	0	0	5 (5)	1 (1)	14
Head	0	0	0	0	197	0	0	0	3
Hips	0	0	0	0	0	191	0	0	9
Rotation anticlockwise	0	1	0	4 (2)	0	0	175	0	22
Rotation clockwise	1	0	0	4 (3)	0	0	0	161	37

Values in table fields are the total numbers of cases from the dataset that were classified to given class. The data come from both low and high-speed datasets

Table 6 Validating the proposed approach (the data comes from both low and high-speed datasets)

Actual condition	Recognition result								
	Clapping	"Not me"	"Jester"	Waving	Head	Hips	Rot-anti	Rot	No recognition
Clapping	90.0 ± 1.6 %	0	0	0	1.0 %	0	0	0	9.0 ± 1.8 %
"Not me"	2.0 %	93.5 ± 1.1 %	0.0 %	1.5 ± 0.4 %	2.5 ± 1.1 %	0	0	0	2.0 %
"Jester"	3.5 ± 0.3 %	0	92.0 ± 0.6 %	0	0	0	0	0	4.5 ± 0.4 %
Waving	0	0	0	93.0 ± 2.2 %	0	0	2.5 ± 1.1 %	0.5 %	7.0 %
Head	0	0	0	0	98.5 ± 0.5 %	0	0	0	1.5 %
Hips	0	0	0	0	0.0 %	95.5 ± 1.0 %	0	0	4.5 ± 0.4 %
Rotation anticlockwise	0	0.5 %	0	2.0 %	0.0 %	0.0 %	87.5 ± 1.7 %	0	11.0 ± 1.4 %
Rotation clockwise	0.5 %	0	0	2.0 %	0.0 %	0.0 %	0	80.5 ± 1.8 %	18.5 ± 1.5 %

Values in the table represent the percentage of the given dataset that was classified to a given class (accuracy) ±SD. The data come from both low and high-speed datasets

4.2 Practical application

After proving that our approach is serviceable, we developed a prototype application that uses a Kinect-based NI with GDL. This application is a three-dimensional desktop that makes it possible to display 3D data acquired by computed tomography (CT). The visualization module is based on our previous work and its detailed computer graphics capabilities have been described elsewhere [31]. The images obtained during tests of the virtual 3D medical desktop prototype are shown in Fig. 8. With this three-dimensional desktop, the user can simultaneously display a number of 3D CT reconstructions (the quantity of data is limited by the size of the graphics processing unit memory) and interact with all of them in real time. The right hand of the user is used for controlling the visualizations. He or she can perform a translation of the volumes (sending all to the back of the desktop, sending them to the front to show details, as in the top right image in Fig. 8), rotate them or control the position of a clipping plane. The last functionality is to change the transfer function (the prototype has three predefined functions: the first shows bones and the vascular system—top left and right pictures in Fig. 8; the second adds extra, less dense tissues to the visualization with a high transparency—the bottom left image in Fig. 8; and the third reconstructs the skin of examined patient—the bottom right picture in Fig. 8). To switch between the translation/rotation/clipping mode, the user makes some predefined gestures with their right hand. These gestures are recognized by GDL.

5 Conclusion

Our approach has proven to be reliable tool for recognizing human body static poses and body gestures. In our approach, static poses—the so-called key frames—form components of dynamic gestures. The recognition rate for all of the tested gestures ranges from 80.5 to 98.5 %. This allows multimedia applications that use our methodology to support the user in a convenient way. We also suppose that if a user becomes familiar with this kind of an NI and learns how to use it, errors in the recognition of a skilled operator may even drop below the measured values. In the future, we are planning to add some functionalities to overcome the current limitations of GDL. The first functionality is the simple ability to recognize movement primitives that requires analyzing the trajectory of joints in a 3D space (like the collinearity of position of particular joints in time). This can be done by adding special functions that would check whether such a condition is present in the memory heap and return the correct logical value to the GDL script rule. The second idea we have come up with is the reverse

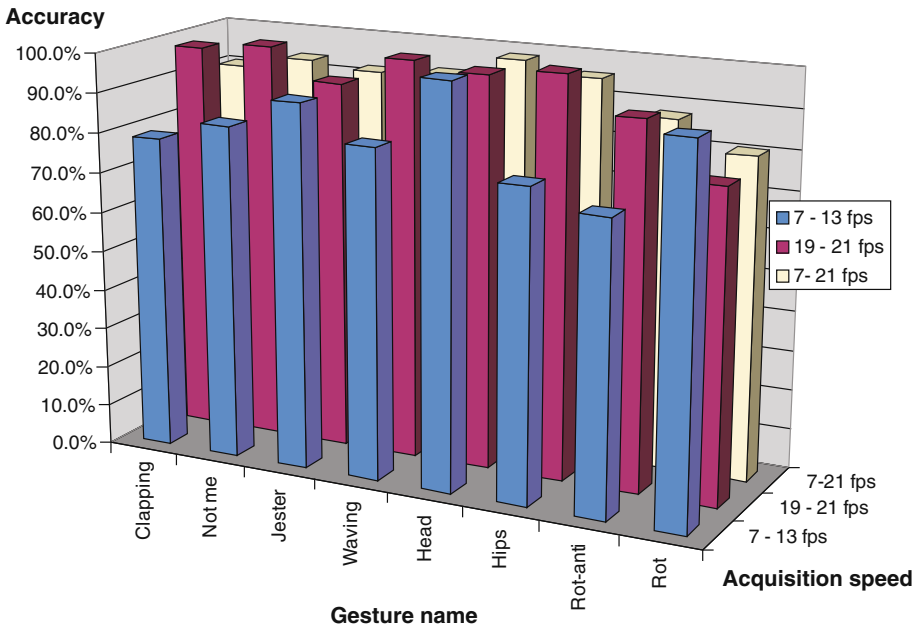

Fig. 7 A plot comparing results from Tables 2, 4 and 6

Fig. 8 Pictures taken during tests of a virtual 3D medical desktop prototype. Detailed description in text

engineering approach. This means developing the ability to generate a GDL description from recorded videos. Such a tool might be very helpful when analyzing the nature of a movement and its characteristic features.

Appendix 1: GDL script specification

GDL is a LALR-1 grammar:

$$GDL = \{V_N, V_T, SP, STS\} \quad (1)$$

Nonterminal symbols, or just nonterminals, are the symbols which can be replaced; thus there are strings composed of some combination of terminal and nonterminal symbols.

$$V_N = \{NumericRule, NumericRule3D, LogicalRule, Rule, GDLScript, Sequence\} \quad (2).$$

Terminal symbols are literal characters that can appear in the inputs to or outputs from the production rules of a formal grammar and that cannot be broken down into "smaller" units.

$$V_T = \{LogicalOperator, NumericOperator, RelationalOperator, OpenBracket, ClosedBracket, OpenSquareBracket, ClosedSquareBracket, LogicalFunction, NumericFunction, SequentialFunction, NumericFunction3D, NumericOperator3D, RuleSymbol, ThenSymbol, Sequence, BodyPart, BodyPart3D, Conclusion, Numeric, Comma, Quotation, Exclamation\} \quad (3)$$

The start symbol of the grammar is:

$$STS = GDLScript$$

All productions that exist in the GDL script are listed in Table 7.

Terminal symbol definition for tokenizer (the terminal symbols scanner):

Table 7 Productions of GDL

No.	Production
1	*GDLScript* → *Rule*
2	*Rule* → *Rule* *Rule*
3	*Rule* → *RuleSymbol* *LogicalRule* *ThenSymbol* *Conclusion*
4	*NumericRule* → *Numeric*
5	*LogicalRule* → *Conclusion*
6	*NumericRule* → *NumericOperator*(″−″) *Numeric Rule*
7	*Numeric3DRule* → *NumericOperator3D*(″−″) *Numeric 3DRule*
8	*NumericRule* → *NumericRule* *NumericOperator* *NumericRule*
9	*Numeric3DRule* → *Numeric3DRule* *NumericOperator3D* *Numeric3DRule*
10	*NumericRule* → *OpenBracket* *NumericRule* *ClosedBracket*
11	*Numeric3DRule* → *OpenBracket* *Numeric3DRule* *ClosedBracket*
12	*NumericRule* → *NumericFunction* *NumericRule* *ClosedBracket*
13	*NumericRule* → *BodyPart* *NumericRule* *ClosedSquareBracket*
14	*Numeric3DRule* → *BodyPart3D* *NumericRule* *ClosedSquareBracket*
15	*NumericRule* → *NumericFunction3D* *NumericRule3D* *ClosedBracket*
16	*Numeric3DRule* → *OpenSquareBracket* *NumericRule* *Comma* *NumericRule* *Comma* *NumericRule* *ClosedSquareBracket*
17	*LogicalRule* → *NumericRule* *RelationalOperator* *NumericRule*
18	*LogicalRule* → *LogicalRule* *LogicalOperator* *LogicalRule*
19	*LogicalRule* → *OpenBracket* *LogicalRule* *ClosedBracket*
20	*LogicalRule* → *LogicalFunction* *LogicalRule* *ClosedBracket*
21	*LogicalRule* → *SequentialFunction* *Sequence* *ClosedBracket*
22	*Sequence* → *Quotation* *SequencePart* *Quotation*
23	*SequencePart* → *SequencePart* *SequencePart*
24	*SequencePart* → *OpenSquareBracket* *Conclusion* *Comma* *Numeric* *ClosedSquareBracket*
25	*SequencePart* → *OpenSquareBracket* *Exclamation* *Conclusion* *Comma* *Numeric* *ClosedSquareBracket*

$NumericOperator = \{+, -, *, /, \%, \wedge\}$

$NumericOperator3D = \{+, -\}$

$RelationalOperator = \{<, >, =, <=, >=, !=\}$

$LogicalOperator = \{\&, |\}$

$LogicalFunction = \{not(\}$

$NumericFunction = \{abs(, sqrt(\}$

$SequentialFunction = \{sequenceexists(\}$

$NumericFunction3D = \{distance(, \text{angle}(\}$

$$\begin{aligned}\text{Body parts} = \{ &\text{ Head, Neck, LeftShoulder,}\\ &\text{RightShoulder, Torso, LeftElbow,}\\ &\text{LeftHand, RightElbow, RightHand,}\\ &\text{LeftHip, RightHip, LeftKnee, RightKnee,}\\ &\text{LeftFoot, RightFoot}\} + \{ \text{.x|.y|.z}\} + \{[\}\end{aligned}$$

$$\begin{aligned}\text{Body parts} = \{ &\text{ Head, Neck, LeftShoulder,}\\ &\text{RightShoulder, Torso, LeftElbow,}\\ &\text{LeftHand, RightElbow, RightHand,}\\ &\text{LeftHip, RightHip, LeftKnee, RightKnee,}\\ &\text{LeftFoot, RightFoot}\} + \{ \text{.xyz[}\}\end{aligned}$$

$Exclamation = \{!\}$

$Numeric = \{number|number1.number2|.number\}$

Conclusion: any string after ThenSymbol is a conclusion. Any unrecognized string is hypothetically a conclusion. Because rules might appear in any order (the previous declaration of a Conclusion after ThenSymbol is not required), at the end of the parsing, the parser checks if all unrecognized strings appear after ThenSymbol. If not, the GDL script contains an error.

The GDL Script also allows one line and multi-line commentaries:

```
//One line commentary
/*
Multi line
Commentary
*/
```

Appendix 2: GDL scripts for the remaining gestures from the validation set

```
//Waving with right hand.
RULE distance(RightHand.xyz[0], LeftHand.xyz[0]) < 200 THEN HandsToClose
RULE RightHand.y[0] < RightHip.y[0] THEN RightHandUnderHeap
RULE abs(angle(RightHand.xyz[0] - RightElbow.xyz[0],
       RightShoulder.xyz[0] - RightElbow.xyz[0]) - 90) <= 20
       & RightHand.y[0] > RightElbow.y[0] THEN WavingGestureCenter
RULE angle(RightHand.xyz[0] - RightElbow.xyz[0],
       RightShoulder.xyz[0] - RightElbow.xyz[0]) > 110
       & RightHand.y[0] > RightElbow.y[0]
THEN WavingGestureRight
RULE sequenceexists("[WavingGestureCenter,1][WavingGestureRight,1]") THEN WavingRight
RULE sequenceexists("[WavingGestureRight,1][WavingGestureCenter,1]") THEN WavingLeft

RULE (sequenceexists("[WavingRight,2][WavingLeft,2]")
       | sequenceexists("[WavingLeft,2][WavingRight,2]"))
       & not(sequenceexists("[HandsReverse,4]"))
       & not(sequenceexists("[RightHandUnderHeap,3]"))
       & not(sequenceexists("[HandsToClose,3]")) THEN Waving
```

$RuleSymbol = \{Rule\}$

$ThenSymbol = \{Then\}$

$OpenBracket = \{(\}$

$ClosedBracket = \{)\}$

$OpenSquareBracket = \{[\}$

$ClosedSquareBracket = \{]\}$

$Comma = \{,\}$

$Quotation = \{''\}$

Note: While performing the Jester gesture, the user is simultaneously waving the left and right hands, so as a matter of fact the Waving gesture is a subset of the "Jester" gesture. To exclude the possibility of an ambiguous recognition, in our GDL script, the memory heap must not contain the *HandsReverse* conclusion for the Waving gesture in the previous 4 s. Our observations showed that a person making the "Not me" gesture may 'accidently wave', so we excluded the *RightHandUnderHeap* conclusion. Also, a wide clapping may be confused with waving, and therefore the *HandsToClose* conclusion was excluded.

```
//Clapping
RULE distance(RightHand.xyz[0], LeftHand.xyz[0]) < 100 THEN HandsTogether
RULE distance(RightHand.xyz[0], LeftHand.xyz[0]) >= 100 THEN HandsSeparate
//exclude jester
RULE sequenceexists("[HandsSeparate,0.5][HandsTogether,0.5][HandsSeparate,0.5]")
       & not(sequenceexists("[HandsReverse,1.5]")) THEN Clapping
Note: where hands are very close to each other the tracking software might misssegmentize
```

Note: Clapping checks if two hands come together and separate within the given time period. When hands are close to each other, the tracking software might segment joints incorrectly—the hand joints might be crossed as in the Jester gesture. Because of that the *HandsReverse* key frame of the Jester gesture was excluded.

```
//Left hand on head
RULE distance(Head.xyz[0], LeftHand.xyz[0]) < distance(Head.xyz[0], Neck.xyz[0])
       & not(sequenceexists("[HandsReverse,4]")) THEN HandOnHead
```

Note: This simple rule checks if the hand is touching the head. Because people might have heads of different sizes, the diameter of the head is estimated as the distance between the joints representing the head and the neck. The hand is very close to the head in the "Jester" gesture, so to exclude the false recognition of the HandOnHead during the Jester, the HandsReverse position must not appear in the last 4 s.

The distance between head and neck joints showed to be good estimator of the overall user size and we used it again in the next rule.

```
//Both hands on hips
RULE distance(RightHand.xyz[0], RightHip.xyz[0]) < distance(Head.xyz[0], Neck.xyz[0])
       & distance(LeftHand.xyz[0], LeftHip.xyz[0]) <distance(Head.xyz[0], Neck.xyz[0])
       & abs(angle(RightHand.xyz[0] - RightElbow.xyz[0],
       RightShoulder.xyz[0] - RightElbow.xyz[0])- 90) < 40
       & abs(angle(LeftHand.xyz[0] - LeftElbow.xyz[0],
        LeftShoulder.xyz[0] - LeftElbow.xyz[0])- 90) < 40 THEN HandsOnHips!
```

Note: This simple rule checks if hands are on hips. People might have different body proportion so size of hips is estimated to be proportional to the distance between joints that represents head and neck.

```
// Rotation of right hand anti-clockwise
RULE HandGoesDown & AnglGoesRight & RightHand.y[0] < RightShoulder.y[0] THEN RotA1
RULE HandGoesUp & AnglGoesRight & sequenceexists("[RotA1,0.5]")
       & RightHand.y[0] < RightShoulder.y[0] THEN RotA2
RULE HandGoesUp & AnglGoesLeft & sequenceexists("[RotA2,0.5]")
       & RightHand.y[0] > RightShoulder.y[0] THEN RotA3
RULE HandGoesDown & AnglGoesLeft & sequenceexists("[RotA3,0.5]")
       & RightHand.y[0] > RightShoulder.y[0] THEN RotationAntiClockwise
RULE sequenceexists("[RotationAntiClockwise,1]")  & not(sequenceexists("[HandsReverse,4]"))
       & not(sequenceexists("[handsUp,4]")) THEN RotationAntiClockwise!
```

Note: This GDL script is very similar to the clockwise rotation but the key frames occur in a different order in the sequences.

References

1. Vinayak, Murugappan, S., Liu, H.R., Ramani, K.: Shape-it-up: hand gesture based creative expression of 3D shapes using intelligent generalized cylinders. Comput. Aided Des. **45**, 277–287 (2013)
2. Zhu, F., Shao, L., Lin, M.: Multi-view action recognition using local similarity random forests and sensor fusion. Pattern Recogn. Lett. **34**, 20–24 (2013)
3. Bickerstaffe, A., Lane, A., Meyer, B., Marriott, K.: Developing Domain-Specific Gesture Recognizers for Smart Diagram Environments, Graphics Recognition. Recent Advances and New Opportunities, pp. 145–156. Springer-Verlag, Berlin (2008)
4. Ray, S.J., Teizer, J.: Real-time construction worker posture analysis for ergonomics training. Adv. Eng. Inform. **26**, 439–455 (2012)
5. Gamage, N., Kuang, Y.C., Akmeliawati, R., Demidenko, S.: Gaussian process dynamical models for hand gesture interpretation in sign language. Pattern Recogn. Lett. **32**, 2009–2014 (2011)
6. López-Méndez, A., Casas, J.R.: Model-based recognition of human actions by trajectory matching in phase spaces. Image Vis. Comput. **30**, 808–816 (2012)
7. Venkatesh Babu, R., Ramakrishnan, K.R.: Recognition of human actions using motion history information extracted from the compressed video. Image Vis. Comput. **22**, 597–607 (2004)

8. Du, Y., Chen, F., Xu, W., Zhang, W.: Activity recognition through multi-scale motion detail analysis. Neurocomputing **71**, 3561–3574 (2008)
9. Elakkiya, R., Selvamai, K., Velumadhava Rao, R., Kannan, A.: Fuzzy hand gesture recognition based human computer interface intelligent system. UACEE Int. J. Adv. Comput. Netw. Secur. **2**(1), 29–33 (2012)
10. Allevard, T., Benoit, E., Foulloy, L.: Fuzzy glove for gesture recognition. In: Proceedings of the 17th IMEKO world congress, pp. 2026–2031, Dubrovnik, June 2003
11. Augsburg University: Full body interaction framework. http://hcm-lab.de/fubi.html (2011)
12. Kistler, F., Endrass, B., Damian, I., Dang, C.T., André, E.: Natural interaction with culturally adaptive virtual characters. J. Multimodal User Interfaces **6**(1–2), 39–47 (2012)
13. Wobbrock, J.O., Wilson, A.D., Li, Y.: Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes, Proceeding UIST '07. In: Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology, pp. 159–168. ACM, New York, 2007
14. Roccetti, M., Marfia, G., Semeraro, A.: Playing into the wild: a gesture-based interface for gaming in public spaces. J. Vis. Commun. Image R. **23**, 426–440 (2012)
15. Kettebekov, S., Sharma, R.: Toward natural gesture/speech control of a large display, EHCI '01. In: Proceedings of the 8th IFIP International Conference on Engineering for Human-Computer Interaction, pp. 221–234. Springer-Verlag, London 2001
16. Chen, Q., Georganas, N.D., Petriu, E.M.: Real-time vision-based hand gesture recognition using Haar-like features. In: Instrumentation and Measurement Technology Conference Proceedings, pp. 1–6. IMTC 2007
17. Arulkarthick, V.J., Sangeetha, D., Umamaheswari, S.: Sign language recognition using K-means clustered Haar-like features and a stochastic context free grammar. Eur. J. Sci. Res. **78**(1), 74–84 (2012). (ISSN 1450-216X)
18. Yeasin, M., Chaudhuri, S.: Visual understanding of dynamic hand gestures. Pattern Recogn. **33**, 1805–1817 (2000)
19. Ruppert, G.C., Reis, L.O., Amorim, P.H., de Moraes, T.F., da Silva, J.V.: Touchless gesture user interface for interactive image visualization in urological surgery. World J. Urol. **30**(5), 687–691 (2012). doi:10.1007/s00345-012-0879-0
20. Clark, R.A., Pua, Y.-H., Fortin, K., Ritchie, C., Webster, K.E., Denehy, L., Bryant, A.L.: Validity of the Microsoft Kinect for assessment of postural control. Gait Posture **36**, 372–377 (2012)
21. Chang, Y.-J., Chen, S.-F., Huang, J.-D.: A Kinect-based system for physical rehabilitation: a pilot study for young adults with motor disabilities. Res. Dev. Disabil. **32**, 2566–2570 (2011)
22. Hachaj, T., Ogiela, M.R.: Recognition of human body poses and gesture sequences with gesture description language. J. Med. Inform. Technol. **20**, 129–135 (2012). (ISSN 1642-6037)
23. Hachaj, T., Ogiela, M.R.: Semantic description and recognition of human body poses and movement sequences with gesture description language. In: Computer applications for bio-technology, multimedia, and ubiquitous city. Communications in computer and information science, vol. 353, pp 1–8 (2012)
24. Schwarz, L.A., Mkhitaryan, A., Mateus, D., Navab, N.: Human skeleton tracking from depth data using geodesic distances and optical flow. Image Vis. Comput. **30**(3), 217–226 (2012)
25. Shotton, F., et al.: Real-time human pose recognition in parts from single depth images, CVPR, p. 3 (2011)
26. Prime Sensor^{TM} NITE 1.3 Algorithms notes, version 1.0, PrimeSense Inc. http://pr.cs.cornell.edu/humanactivities/data/NITE.pdf (2010)
27. Zhang, Q., Song, X., Shao, X., Shibasaki, R., Zhao, H.: Unsupervised skeleton extraction and motion capture from 3D deformable matching. Neurocomputing **100**, 170–182 (2013)
28. Liu, Y., Huang, Q., Ma, S., Zhao, D., Gao, W.: Joint video/depth rate allocation for 3D video coding based on view synthesis distortion model. Signal Process. Image Commun. **24**(8), 666–681 (2009)
29. Khoshelham, K.: Accuracy analysis of Kinect depth data. In: Lichti, D.D., Habib, A.F. (eds.) ISPRS workshop laser scanning 2011. International Society for Photogrammetry and Remote Sensing (ISPRS), Calgary, August 2011
30. Khoshelham, K., Oude Elberink, S.J.: Accuracy and resolution of Kinect depth data for indoor mapping applications. Sens. J. Sci. Technol. Sens. Biosens. **12**(2), 1437–1454 (2012)
31. Hachaj, T., Ogiela, M.R.: Visualization of perfusion abnormalities with GPU-based volume rendering. Comput. Graph. **36**(3), 163–169 (2012)