Revista Romana de Interactiune Om-Calculator 11(2) 2018, 163-183 © MatrixRom

A Knowledge-Based Model for Human Motion Tracking and Gesture Recognition in the Context of Natural Interaction with Kinect Devices

Sabin C. Buraga, Elena Creangă

Faculty of Computer Science, Alexandru Ioan Cuza University of Iasi, Romania
Berthelot, 16, 700483, Iasi
E-mail: busaco@info.uaic.ro, elena.creanga.mail@gmail.com

Abstract. Human-computer interaction (HCI) was revolutionized by the emergence of new technologies, be they proof of concept or fully functional, which created a more immersive, integrated and interactive experience, interactive systems being now able to understand user's activities and respond accordingly. Our study presents an original knowledge-based model to be used to semantically describe – in terms of the OWL and RDF standards – human gestures and poses tracked by a motion sensor-based device. To prove the feasibility of the proposed conceptualization, two use cases were designed: one for manipulating 3D shapes on a board and one for usability testing by using different Kinect-based applications.

Keywords: natural user interaction, knowledge modeling, human gesture, Kinect, experiments.

1. Introduction

As technology advances, it is important that it also continuously immerses into humans' everyday life, changing the way society interact and perceive it, up to the point that one is no longer aware of its existence. Nowadays, people are looking for ways to control their environment without being restrained to using a traditional interaction paradigm – based on mouse and keyboard, only – and, why not, not even a smart phone. In this context, new interaction paradigms and technologies have been developed, be they proof of concept or fully functional. From motion tracking to brain weaves reading, they all try to tackle the idea of an immersive smart environment, helping the user naturally interact with everyday objects.

Consulting the scientific literature, we observed the lack of a proper knowledge-based model able to specify complex natural interactions, by representing sensor information, motion tracking, and gestures. Our aim is to specify a conceptualization of main aspects concerning these natural interactions – discussed in Section 2 of this article. To accomplish this goal, the actual standards of semantic Web are used to denote body poses, user gestures, sensor information and others – see Section 3.

Several use cases are also described in Section 4, by using the Microsoft Kinect (Han *et al.*, 2013) – a motion tracking device that introduced a major shift in the way we interact with technology, by converting the whole environment into a canvas, with users' moving body acting as a stylus.

The paper ends with conclusions and further directions of research.

2. Natural Interactions: Motion Tracking and Gesture Recognition

2.1 Motion Tracking

Motion tracking (Yilmaz, Javed & Shah, 2006) represents the ability to track the motion of some object over time. In the most often cases, this object is the human body, but the tracking is not restricted only to it. In essence, motion tracking requires identifying particular points or shapes or objects, and tracking their individual movements. The difficulty of motion tracking consists of identifying a specific object that is being tracked, which implies first detecting movement in general. Once an object that is being tracked is recognized, the goal of motion tracking is to determine in which direction the object is moving, usually in a three-dimensional space. If a particular object has been identified and its movement is tracked, then the gathered data can be used to perform actions such as controlling a cursor on a screen, or it can be examined over time to detect certain gestures, which are useful for creating a language to communicate with a specific software application.

To perform our research, the Microsoft Kinect device was selected. Kinect has already a large number of practical usages – for example, those described by Lun & Zhao (2015) and Webster & Celik (2014).

Using its range of sensors, Kinect can track up to six people at the same time, constructing skeletal tracking – i.e. tracking of various body parts – for each of them individually. While in full skeleton tracking mode, 25 joints are being recognized, whereas while in seated mode, only 12 are analyzed. These joints include hand tips and thumbs. The API (Application Programming Interface) provides access to various sensors through its frame readers, significant data being provided by *Body* and *BodyIndex*.

Also, for further processing additional data could be gathered via *Color*, *Depth*, *Audio*, *Infrared*, and *LongExposureInfrared* sensors[14].

The skeletal body tracking – depicted in Figure 1 – is obtained by using a *BodyFrame* item, accessed from *BodyFrameReader*. The *BodyFrame* represents a set of 25 joints' positions, for each of the maximum of 6 persons that can be simultaneously tracked. Each joint is placed on the 3D Cartesian coordinate system $< X, Y, Z >$, having a specific orientation and is assigned a specific number to identify it.

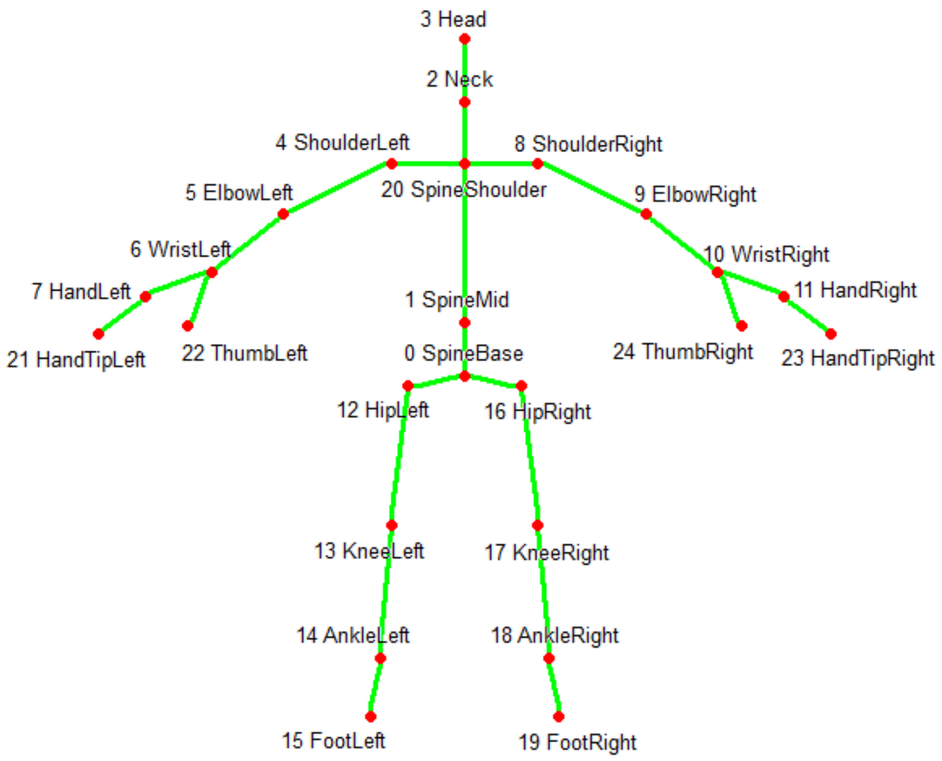

Figure 1. Human body joints tracked by a Kinect device.

[14] Microsoft Kinect – https://developer.microsoft.com/en-us/windows/kinect

2.2 Gesture Recognition

The *gesture* is a specialized term for a whole range of different disciplines, which make it buckle under its weight various overlapping and sometimes conflicting meanings ascribed to it. From the human computer interaction point of view, “a gesture is a motion of the body that contains information” (Laurel & Mountford, 1990). For example, waving goodbye or rotating a picture are considered gestures. According to Mitra & Acharya (2007), *gesture recognition* represents the foundation for developing applications using natural user interfaces, in order to provide support for actions like navigation, data input, performing various tasks, etc.

In the Kinect context, gestures are defined as the relative positions of some specific body joints for a given number of frames. By default, Kinect offers only three predefined gestures, as states for hands: *Open*, *Closed*, and *Lasso*. Gestures can be both static, consisting of a single state, such as pointing, dynamic and continuous, as a sequence of multiple parts, with each part of a gesture as a specific movement. Therefore, all these segments combined will then form the whole gesture. Taking the wave gesture as an example, it happens when users raise one of their hands above the corresponding elbow, moving it from side to side, with the elbow being still. This means the base of the wave gesture would consist of two segments with the following states: “hand above the elbow and hand to the right of elbow” and “hand above elbow and hand to the left of elbow”.

However, recognizing segments is not enough, if a degree of confidence is not taken into account. For instance, in the case of the wave gesture, if the hand would drop right after completing the segments tracking its movement from the left to the right of the elbow, it would still be matched as a wave gesture, even if that was not the intended purpose. Thus, in order to acquire the degree of accuracy when tracking it, we require the user to repeat the segments three times and wait for the successful completion of all.

Offering the tracking for the 25 joints is all that the Kinect does, with no further conceptualization of the human body. For an accurate measurement of a rather simple gesture such as waving, we need to compare every time the two joints – *Elbow* and *Hand* – on all three axes. It would be so much easier if we could manipulate them using a more natural manner (i.e. relative natural language constructs) instead, having relative comparisons between the two.

To accomplish this goal, a knowledge-based model should be envisioned in order to specify the structure of tracked human body.

3. A Knowledge Model for Representing Natural User Interactions

In order to properly recognized gestures and perform associated tasks, the main components of the human body should be to formally specified as formal components of the desired conceptual model.

3.1 Considerations regarding Body Poses

First step is to define the joints as more than simple enumerators, but as objects which can be compared – from a point of view of relative positioning in the Cartesian space – one to another using comparisons similar to natural language constructs as follows:

- "Above", "Below",
- "After", "Before",
- "ToTheLeftOf", "ToTheRightOf",
- "AtTheSameLength", "AtTheSameHeight", "AtTheSameDepth",
- "OverlapsWith".

Next, these joints are merged into segments, to specify certain *body poses*. This results in having a bone-like structure, from hands to feet which permits identifying the pointing direction for each of them. Because there are some poses, such as both arms raised above the head – the equivalent of the "surrender" action –, we also provide the possibility to tie together mirroring segments as one entity.

A *pose* represents the most fine-grained part of a *gesture* – $G \subseteq P$, where $P = \{ p_i, i = 1,...,N \}$ –, as it can be defined as small as just a simple comparison between two joints. However, the whole purpose of this conceptualization is to offer an ease of use in manipulating and creating gestures. Therefore, the possibility of combining multiple atomic parts is important.

Consequently, the following logical constructs are provided:

- Merging two poses – "and" operation defined by the "&" symbol.
- Assuming the correctness of only one – "or" denoted by the "|" symbol.
- Matching the opposite of a pose – "not" logical operator specified by the "!" symbol.

For example, a given pose $p \in P$ could be considered as a composition of three sub-poses: $p = p_1$ & $(p_2 \mid ! \, p_3)$.

From the computational point of view, in order to determine a joint's position relative to another one, it is necessary to determine the length between them in 3D space. However, as the distance between the wrist and the hand is smaller than between hand and elbow, the difference in height and angle between the two should be proportional, resulting in the same confidence rate when comparing both. For maintaining this proportion, we needed to calculate the shortest distance between any two given joints and add the positions of each intermediary joint to the final ratio.

To investigate the angle between two interconnected segments, there was also necessary to define the concept of *limbs*. From the biological perspective, limbs are defined as the peripheral parts of the human body – arms and legs, more precisely. In this particular case, we considered as limbs the connecting bones between hip and ankle (as feet) and the ones between shoulder and wrist (as arms). This allowed to analyze and combine the rotation and angle for hands and feet.

A model of the pose we want to achieve is being created and then it is compared to the actual position of the body from a Kinect frame, resulting in a degree of confidence that the two are similar. Correctly identifying the pose is vital, as we would not want to misinterpret the user's intention and execute an unwanted, possibly irreversible action.

To avoid receiving false positives, we only considered as valid matchings of over 80%.

3.2 Aspects concerning Gestures

As specified by Kinect guidelines, gestures can belong to three types: static, dynamic, and continuous. Since the difference between dynamic and continuous is in the way each interacts with the application, it has not been the subject of modeling it. Instead, we represented gestures by a combination of segments which are performed in a specific order in a certain time frame – formally, the gesture set is $G = \{ g_i, i = 1,...,M \}$.

Each segment is described by a range of poses, depicting the positions of certain joints or bones. Therefore, we can create both static gestures – such as crossed arms, which is formed out of one single gesture segment – and, at the same time, we can create a model for more complex gestures – for example, wave or clap. Yet, when it comes to more complex gestures, the more complex they are, the trickier they are to detect, as the exact sequence of segments must be performed by the user in the exact same order. States for each gesture segment exist precisely for preventing early dismissal of a gesture as a whole.

Considering the wave gesture as an example, the hand would have to swing from the left of elbow to its right. If, for some reason, the captured frame is not conclusive of whether the hand moved from the left side to the right side (meaning it still detects the hand as being lifted and open, but not certain about the position relatively to the elbow), it will wait another F frames – for practical considerations, we chose $F = 10$ – to reanalyze the same segment matching, keeping the validity of the previous ones, by yielding the outcome of the current evaluation as UNDETERMINED.

If any of the base conditions (the hand must stay up and open) are not met, then the result will be considered as FAILED, reverting all previous segments' validity.

3.3 KONTO: An Ontology for Representing Sensor Information and Gestures

To model the user movements, including poses and gestures, and the interaction with a Kinect device, an ontology – named KONTO – was designed. This conceptual proposal is able to represent both sensor information and gesture-based interactions. The conceptualization is expressed in OWL – Web Ontology Language (Allemang & Hendler, 2011), a well-known W3C standard for expressing knowledge on the Web based on the < *subject*, *predicate*, *object* > triples defined by the RDF (Resource Description Framework) model.

To create the terminology – TBox (Baader *et al.*, 2003) logical component – of the knowledge base, the following classes are defined to mainly denote:

- Data provided by a device (in this case, Kinect): *Sensor* class.
- General information regarding a user (*User*), plus observed anatomical body parts and joints: *Body*, *BodyPart*, *Part*, *Limb*, *Arm*, *Leg*, *Bone*, and *Joint* classes.
- Detected gestures and poses: *Gesture*, *GestureSegment*, *HandState*, and *Pose* classes.

Main classes of the KONTO ontology are depicted in Figure 2.

Several details are presented in the next subsections.

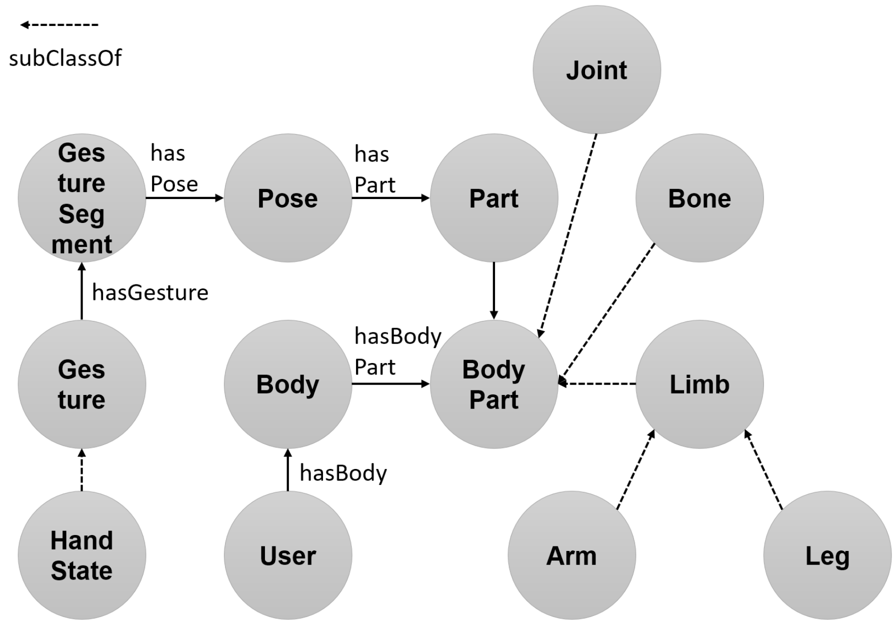

Figure 2. KONTO – main concepts.

Sensor

This class represents the generic sensor which is used to identify the user. The object property *detectsUser* could be used to define the relation between a *Sensor* instance and a detected user.

Further knowledge about a user could be expressed by FOAF (Friend Of A Friend) ontology (Graves, Constabaris & Brickley, 2007) or the *Person* class provided by the schema.org specification (Guha, Brickley & Macbeth, 2016), both well-known conceptual models.

Body

The class is identified using the *trackingId* functional property, having as a value a RDF literal. Also, it can include multiple instances of the *BodyPart* class, specified with the property *hasBodyPart*.

At a syntax level, this could be denoted by the following RDF statements – expressed in Turtle[15] format – which are included in the assertion

[15] RDF 1.1 Turtle. Terse RDF Triple Language (W3C Recommendation, 2014) – https://www.w3.org/TR/turtle/

component (ABox) of the defined knowledge base:

```
Body rdf:type owl:Class .

trackingId rdf:type owl:DatatypeProperty ,
                    owl:FunctionalProperty ;
  rdfs:domain Body ;
  rdfs:range  rdf:Literal .

hasBodyPart rdf:type owl:ObjectProperty ;
  rdfs:domain Body ;
rdfs:range  BodyPart .
```

The *BodyPart* class could represent some elements of the human body and is a class parent of the *Limb*, *Bone*, and *Joint* sub-classes.

In the case of *Limb*, this class is a parent for *Arm* and *Leg* sub-classes and has at least one instance of the *Bone* class, defined through property *hasBone*. The *Arm* class has only two disjoint individuals belonging to the *RightArm* and *LeftArm* classes. Similar constructions are made for the *Leg* class.

The *Bone* class is a sub-class of *BodyPart* and is formed of exactly two instances of the *Joint* class defined through property *hasJoint* of cardinality 2 (each bone have only two joints).

Joint

This class specifies the human body joints and is a sub-class of *BodyPart*. Also, it has 25 individuals defined, according to the model used by a Kinect device discussed in Section 2.1 and depicted by Figure 1.

Part

This class denotes the tie between two instances of the *BodyParts* and an action (specific to a certain context, platform, application, and/or paradigm of interaction). The action could be defined by external conceptual models such as the *Action* class defined by *schema.org* model (Guha, Brickley & Macbeth, 2016).

Additionally, the class has useful properties defined in relation to itself: *isAbove*, *isBelow*, *isAtTheSameHeightOf*, *isAtTheSameLengthOf*, *isAtTheSameDepthOf*, *isToTheRightOf*, *isToTheLeftOf*, *overlapsWith*.

These properties could be useful to perform further processing (in this case, reasoning) in order to detect complex gestures and/or poses, depicting the relationships between different instances of the *BodyPart* class. An example: *RightHand totheLeftOf RightElbow* – this RDF triple specifies that the right hand of the tracked user is placed to the left of his/her right elbow.

Pose

The class represents a union of instances of the *Part* classes, defined through the object property *hasPart*, and is used to specify the relationship between different parts of the user's body.

For example, in order to describe a specific pose, following two RDF constructs are generated:

```
aPose rdf:type Pose ;
      hasPart RightHandToTheLeftOfRightElbow ;
      hasPart RightHandAboveRightElbow .
```

GestureSegment

In addition, the class *GestureSegment* embodies a multitude of poses defined with the *hasPose* property and is used to illustrate the connection between poses such as:

```
aGestureSegment rdf:type GestureSegment ;
  hasPose RightHandTopLeftOfRightElbow .

anotherGestureSegment hasPose
  LeftHandBelowAtTheSameLengthWithLeftElbow .
```

Gesture

This class denotes a gesture and has defined at least one instance of the *GestureSegment* class, by using the *hasGestureSegment* object property.

4. Use Cases and Experiments

In order to demonstrate the potential of the concept presented, several use cases were necessary to be analyzed. For this purpose, we chose to envision two projects: one regarding the manipulation of shapes in 3D space and one concerning the usability testing of the Kinect-based applications.

Similar experiments are described by Kim *et al.* (2016), Sridhar *et al.* (2016), Grădinaru & Moldoveanu (2016), and Mocanu *et al.* (2016), but they are not focused on using the ontological models.

4.1 Shape Manipulation in 3D Space

The easiest way to test the validity of the model was by having direct visual of how the user can control and interact with an application.

Therefore, we chose a representation of shapes in 3D as a playground considered an interactive virtual zone of natural interaction.

Initially, the board presents itself filled with three shapes, rendered using OpenGL[16]: a sphere, a cube and a pyramid. This playground for 3D shapes represents the first proof for the ease of modeling gestures using the above presented conceptualization.

Since the main manipulators of our shapes are the hands, we want to keep track of every move they make. Consequently, we also represent them on the board, as spheres. For better visualizing their position on the *X*—*Y*—*Z* axes, lines are also drawn from origin in each direction, for each hand. We also display the raw frames captured by the sensor to better map the interaction. Moreover, as soon as the body is starting to be tracked, a skeleton overlay shows up directly on the image, to have a better sense of how the body is being perceived by the sensor – see Figure 3.

The mapped behavior of the drawing board is represented by both static, and dynamic gestures which should be detected and further processed by our developed library.

[16] Open GL ES – https://www.khronos.org/opengles/

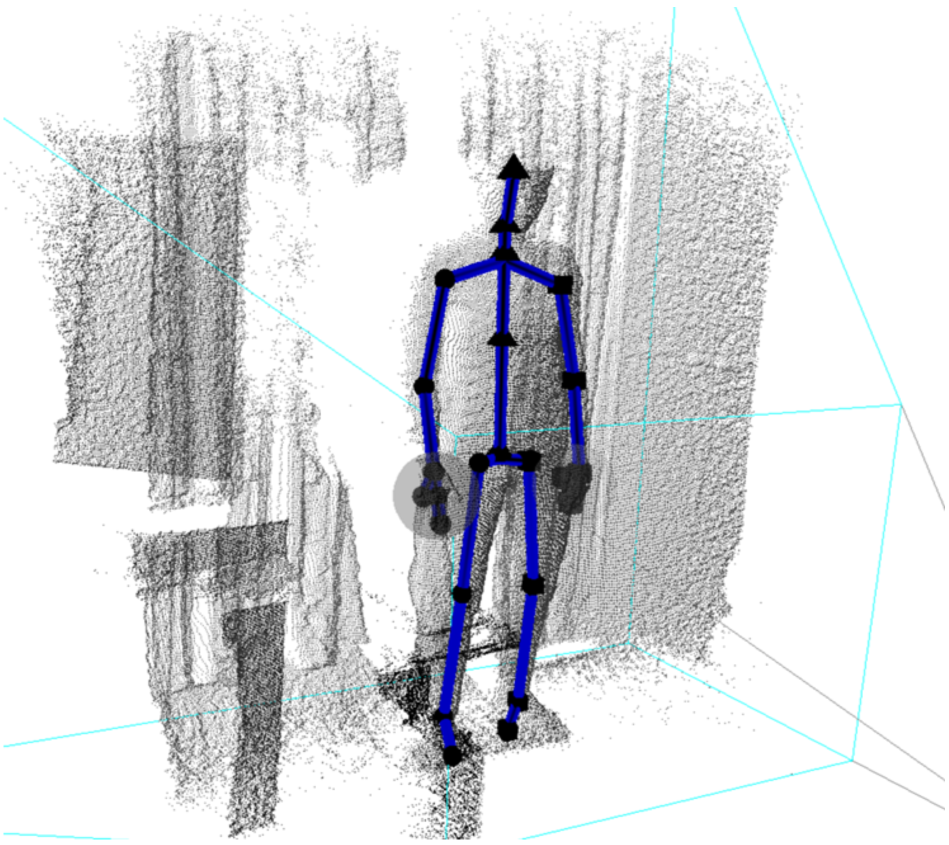
Figure 3. Tracking the user's pose: shape module.

Crossed Arms Static Gesture

For instance, the CROSSED ARMS static gesture is depicted by having one segment kept for 30 frames. This segment consists of having the right hand overlapping the left shoulder, the left hand overlapping right shoulder and both elbows below hands – these constraints are also checked by using the RDF assertions generated by the tool, according to the conceptual model described in Section 3. When only one hand is detected to be in the right position, the gesture is not entirely dismissed, but declared UNDETERMINED, in the wait for the second hand to come to the desired position.

This gesture is linked to the "X" symbol which is associated with closing or deleting a program or activity. In this particular case, when CROSSED ARMS gesture is detected, the drawing board erases all its shapes.

Wave Dynamic Gesture

The WAVE dynamic gesture is modeled using two segments, repeated three times: one segment for detecting the movement of the hand to the right of elbow and one for detecting the movement to the left of elbow. Each segment consists of two poses as follows: one for determining that the hand is above the elbow and one for determining the hand's position on the *X*-axis in rapport with the elbow.

While the hand remains still up, but the confidence of the movement on the *X*-axis is yet below the 80% imposed threshold, the outcome of the segment evaluated is going to be UNDETERMINED. If the user decides to drop the hand, the outcome of both current segment and total gesture will become FAILURE (according to the discussion from Section 3.2). A successfully completing the two-way movement, three times in a row, will result in successfully recognizing it – i.e. the DONE state.

The waving gesture stands both for salutation and for erasing a drawing board. Combining the latter with the already familiar shake gesture necessary for undoing written text on the iOS devices, we mapped this gesture to undoing the previous movement on the 3D board.

Clap Dynamic Gesture

The CLAP dynamic gesture is formed of two segments, repeated two times: one for joining the hands in the middle of the torso, and one for separating them to the sides of it. While the joining hands segment starts by making sure the hands are in front of the shoulders, then that they both are between shoulders and hips and finally, that they overlap one another, the second one is represented the opposite way.

The CLAP gesture is already familiar to the user for signaling the beginning of a new sequence of actions (for instance, those provided by the Dance Central application) or a new game (e.g., Kinect Sports). Therefore, when detecting it, the board will reset to its initial state.

Grab Action

For manipulating the objects on the playground, one needs to position the right hand on one of the objects and then grab it. Having the right hand closed, the user can move the shape similarly to the movement of an object in real life. While the right hand is still in grabbed position, having the left hand closed as well, will determine rotating and zooming of the current object.

4.2 Usability Testing Experiments

Movement tracking and gesture detection are used in all applications which are destined for the Kinect motion sensor. The model we created can be extrapolated and used for developing any of them.

Currently, every application has its own – better or worse – manner of detecting gestures. Gestures are all arbitrary and this implies that, depending on the context, each gesture has a different meaning, which is more or less intuitive.

What is signified as a gesture is in fact an intent of having an action executed. Movements have no meanings outside the ones we ascribe to them. Pointing and shrugging are the only two gestures which anthropologists have found as universal. While the first one has a generally agreed upon use, the second is too subtle to identify.

Any other gesture must be based on an agreement between the designers of an application and its users. Therefore, the designers must either teach the users the significance of the gestures or they must depend on pre-established conventions (Webb & Ashley, 2012).

This is why, for the second use case presented below, we designed a module meant to analyze the user's interaction with different applications, in order to *detect the flow in which the users are engaging*. All gathered data is stored as RDF documents, by using the conceptualization presented in Section 3, in order to be further processed.

We chose for the conducted experiments the following two Kinect applications:

- A virtual dressing room – Fitnect.
- A Mathematics educational game – Jumpido.

Fitnect

Fitnect[17] is a virtual store which uses augmented reality (AR) to create a 3D fitting room interactive environment. This allows users to try on different clothes and accessories without actually having to wear them physically. Since its main objective is to facilitate the ease of trying on clothes, the gestural interaction should not be too complex.

[17] Fitnect, a 3D Virtual Fitting Room – http://wiki.fitnect.hu/

The application begins with a silhouette in the center of the screen and a circle surrounding its feet, as a stage. Even though it can easily fully detect a silhouette, it does not start right away, requiring the user to be in a "surrender" pose.

We specified the mapping of this pose by using the bone structure, defined as the connection between two neighboring joints, in order to be able to analyze the angle formed by the clavicle and elbows and forearms and shoulders. Therefore, the "surrender" pose is achieved when the right elbow is to the right of right shoulder, left elbow is to the left of left shoulder, the forearms together with shoulders' joints form an angle of 90 degrees on each side and upper arms and clavicle form a straight angle.

The movement of lifting one leg at a time was modeled as a combination between positions and angles, consisting of two gesture segments. More precisely, when checking the match for the gesture of lifting right leg, the first segment is checking whether the angle formed by the right shin and right hip is either obtuse or right, the right knee is in front of spine base joint. Because we had to differentiate between kicking and lifting the leg, an additional condition had to be fulfilled: the right ankle must not be situated before the right knee. To avoid identifying a squat as a lifting of the legs, the left knee had to be at the same depth as spine base. The lifting of the left leg was achieved mirroring the behavior described for the right one.

Due to the fact that the accessories and try-ons were all accessed through hovering the hands over certain positions on the *X*—*Y* axis, corresponding on the screen, we were unable to track whether the users would lift their hands to analyze the piece of clothing or in order to change the outfit, therefore we excluded it from tracking.

To assess the application's usability, a user testing experiment was conducted following the processes described in (Negru & Buraga, 2012).

From our observations, the first-time users of the application that we tracked had some difficulties figuring out the trigger for starting the try-on, managing to conform to the position in the first 30 seconds, representing only 16% of all 104 interaction gestures from all 12 involved persons.

The choosing of an outfit took another few seconds and, afterwards, the users intuitively lifted the legs to see if the outfit had any reaction to that. The lifting of the right leg had the biggest percentage – 43%, whereas the left leg only 38%. The wave gesture was also detected, with 3%, which could mean the feeling of lack of application's responsiveness. The average interaction of one user lasted 90 seconds.

Jumpido

Focused on children as target audience, Jumpido[18] is an application meant to complement teacher's activities in Mathematics classes. It embeds a series of mini games which teach the children notions such as sorting, addition, subtraction and percentages. Each game's controls differ, from simple hover and hold actions over predefined areas to more complex moves, such as jumps, squats and kicks.

For our experiment, we chose to analyze a game with a familiar theme: football. We can control the game using jumps, squats and kicks using any leg. Each round consists of a question and a series of answers, through which we must navigate through by either squatting (move to the answer below) or jumping (move to the answer above). Marking a result as correct is done by kicking the football into the net with either leg.

As we mentioned above, all detected gestures were specified and annotated by using ontological assertions expressed in OWL/RDF according to the conceptual model described in Section 3.3.

For each detected gesture of a given user, a RDF dataset is generated by using additional conceptual models like FOAF, schema.org and Usability Testing extensions proposed by Negru & Buraga (2012). An excerpt is presented below:

```
    usabilityTest1 rdf:type ut:UsabilityTest ;
      ut:participant user1 , ..., user12 ;
      ut:userTask [ ut:taskType "gesture" ;
                    ut:taskResult "kick" ;
                    ut:action kick ] ;
      ut:successCriteria "0.35" .

    kinectBodySensor rdf:type Sensor ;
                     detectsUser user3 .

    user3 rdf:type foaf:Person .

    kick rdf:type Gesture .
...
```

[18] Jumpido, a series of educational games for primary school Mathematics – http://www.jumpido.com/en

The generated RDF datasets could be further processed by using various machine learning techniques (Ibanez *et al.*, 2016; Nguyen & Le, 2015; Setiawan *et al.*, 2017). For example, a study of interest can be focused on predicting the user's next gesture or classifying users based on recurrent gestures and body poses performed for a class of software applications. Another idea is to use a fuzzy logic approach to tackle the uncertainty of human behavior (Diaz-Rodriguez *et al.*, 2014).

The concept of *kick* is represented by two segments: kicking and returning to normal standing pose. Similar to lifting of the leg, kicking has an important differentiator in the foot's depth compared to other joints. Therefore, besides analyzing the depth of the knees versus spine base and the angles between shins and hips, and respectively thighs and spine base, we must also check that the foot of the leg that is producing the kick has the smallest depth of all the joints in the body.

While detecting a waving hand is rather straightforward, gestures such as jumps and squats are more complex, as they can be easily mixed up with other segments from different movements.

The *squat* movement consists of two segments, one for expressing the position of the body when squatting and another one for depicting the returning to normal standing pose. Therefore, the first one analyzes the angles between shins and hips (acute or right), between the thighs and spine base (straight or obtuse). The second one checks whether the above mentioned angles change back to the ones corresponding to normal standing position or not and that the depth of all lower body joints is the same.

Jumping is besides analyzing body joints, also about considering various parameters from the human body and the environment. When modeling it, we split it into four segments: slight bending, jumping and returning to normal standing position. The first one measures the angles between shins and hips (obtuse), thighs and spine base (obtuse) and the depth distance between knees and spine base. The second segment checks whether the shins are in straight angles with the hips and measures the height of the user as well as the position of the head joint. The third one analyzes the angle between shins and hips (straight) and thighs and spine base (right) as well as comparing the user's height and the position of the head joint with the ones captured in the second segment to be the same and lower respectively.

In the case of gestures performed by children while playing a Jumpido module, we noticed the following:

- As the beginning of the game is marked by a single jump, it is visible that it is the first gesture registered. However, as the gestures necessary for interacting with the application are presented in a visual manner in the beginning of the module, we remarked that the subjects figure what they have to do from as early as the first 30 seconds of interaction.
- The majority of users were right handed, since 66% of the kicks were performed using the right foot, whereas only 34% with the left foot.
- Out of all 261 detected gestures for 9 involved users, 28% were jumps, 37% were squats, 23% were kicks with the right foot, and 12% were kicks with the left foot.

A snapshot regarding the participants of the conducted usability test is presented in Figure 4.

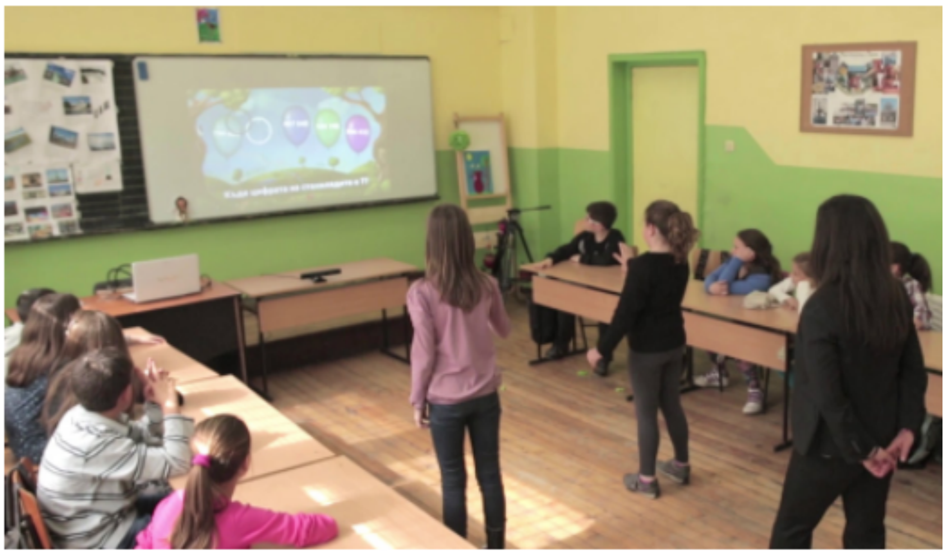

Figure 4. Conducted usability test on primary school pupils.

5. Conclusion

Motion tracking and gesture recognition techniques allow designers and HCI engineers to depict new ways of interacting with computers, blurring the line between imagination and reality. The natural interaction between current devices and users opens new possibilities in the realm of ubiquitous computing and computer vision, by allowing ease of access of applications through posture analysis.

The current paper proposed an original conceptual model to be used in gesture-based interactive applications. We created the KONTO ontology with the purpose of defining, annotating, and interlinking data and knowledge modeling postures and gestures in an easier way, by using the existing semantic Web standards (RDF and OWL) and various other well-known vocabularies, with respect to the Linked Data principles (Bizer, Heath & Berners-Lee, 2008).

While the conceptual model we created was designed having Microsoft Kinect as its main focus, it can also be easily extended to other tracking devices. The KONTO model could be also used in conjunction to other knowledge-based solutions for motion capture (Woodcock, 2015) or Web of Things platforms (Raggett, 2015).

We tested the usefulness of our conceptual approach by creating a library used for creating a playground for manipulating 3D shapes and a usability testing tool for analyzing several Kinect-based applications.

Based on observed movements and gestures performed by users, our near-future research intention is to analyze data in order to automatically create (proto-)personas by following the dynamic method proposed by Pichot & Bonnardel (2018).

References

Allemang, D., Hendler, J. (2011) *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*. Elsevier.

Baader, F., Calvanese, D., McGuinness, D., Patel-Schneider, P., Nardi, D. (2003) *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press.

Bizer, C., Heath, T., Berners-Lee, T. (2008). Linked Data: Principles and State of the Art. *World Wide Web Conference (WWW 2008)*. ACM, 1–40.

Diaz-Rodriguez, N., Cadahia, O. L., Cuéllar, M. P., Lilius, J., Calvo-Flores, M. D. (2014) Handling Real-World Context Awareness, Uncertainty and Vagueness in Real-Time Human Activity Tracking and Recognition with a Fuzzy Ontology-based Hybrid Method. *Sensors*, 14 (10), 18131–18171.

Grădinaru, A., Moldoveanu, A. (2016) Kinect v2 Evaluation for In-home Medical Rehabilitation Scenarios. *Romanian Journal of Human-Computer Interaction*, 9 (1), 1–

18.

Graves, M., Constabaris, A., Brickley, D. (2007) FOAF: Connecting People on the Semantic Web. *Cataloging & Classification Quarterly*, 43 (3–4), 191–202.

Guha, R. V., Brickley, D., Macbeth, S. (2016) Schema.org: Evolution of Structured Data on the Web. *Communications of the ACM*, 59 (2), 44–51.

Han, J., Shao, L., Xu, D., Shotton, J. (2013) Enhanced Computer Vision with Microsoft Kinect Sensor: A Review. *IEEE Transactions on Cybernetics*, 43 (5), 1318–1334.

Ibanez, R., Soria, A., Teyseyre, A. R., Berdun, L., Campo, M. R. (2016) A Comparative Study of Machine Learning Techniques for Gesture Recognition using Kinect. *Handbook of Research on Human-Computer Interfaces, Developments, and Applications*. IGI Global.

Kim, J., Jung, H., Kang, M., Chung, K. (2016) 3D Human-Gesture Interface for Fighting Games using Motion Recognition Sensor. *Wireless Personal Communications*, 89 (3), 927–940.

Laurel, B., Mountford, S. J. (1990) *The Art of Human-Computer Interface Design*. Addison-Wesley.

Lun, R., Zhao, W. (2015) A Survey of Applications and Human Motion Recognition with Microsoft Kinect. *International Journal of Pattern Recognition and Artificial Intelligence*, 29 (5).

Mitra, S., Acharya, T. (2007) Gesture Recognition: A Survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37 (3), 311–324.

Mocanu, I., Marian, C., Rusu, L., Arba, R. (2016) A Kinect based Adaptive Exergame. *12th International Conference on Intelligent Computer Communication and Processing (ICCP 2016)*, IEEE, 117–124.

Negru, S., Buraga, S. (2012) A Knowledge-Based Approach to the User-Centered Design Process. *International Joint Conference on Knowledge Discovery, Knowledge Engineering, and Knowledge Management*. Springer, 165–178.

Nguyen, D., Le, S. (2015) Kinect Gesture Recognition: SVM vs. RVM. *Seventh International Conference on Knowledge and Systems Engineering (KSE 2015)*. IEEE, 395–400.

Pichot, N., Bonnardel, N. (2018) Enhancing Collaborative Creativity: Towards a New User-Centered Design Method, the Dynamic Persona Method. Proceedings of the 20th Congress of the International Ergonomics Association (IEA 2018). Springer, 580–591.

Raggett, D. (2015) The Web of Things: Challenges and Opportunities. *Computer*, 48 (5), 26–32.

Setiawan, F. A., Budiardjo, E. K., Basaruddin, T., Aminah, S. (2017) A Systematic Literature Review on Combining Ontology with Bayesian Network to Support Logical and Probabilistic Reasoning. *Proceedings of the 2017 International Conference on Software and e-Business*. ACM, 1–12.

Sridhar, S., Bailly, G., Heydrich, E., Oulasvirta, A., Theobalt, C. (2016) Full-Hand: Markerless Skeleton-based Tracking for Free-Hand Interaction. *MPI-I-2016-4-002*. Max

Planck Institute for Informatics: Saarbrücken, Germany.

Yilmaz, A., Javed, O., Shah, M. (2006) Object Tracking: A Survey. *ACM Computing Surveys*, vol. 38, no. 4.

Webb J., Ashley, J. (2012) *Beginning Kinect Programming with the Microsoft Kinect SDK*. Apress.

Webster, D., Celik, O. (2014) Systematic Review of Kinect Applications in Elderly Care and Stroke Rehabilitation. *Journal of Neuroengineering and Rehabilitation*, 11 (1).

Woodcock, R. (2016) Capture, Hold, Release: an Ontology of Motion Capture. *Studies in Australasian Cinema*, 10 (1), 20–34.