MASTER’S THESIS

A conceptual model of user-based motion tracking

Scientific coordinator

Conf. Dr. Sabin-Corneliu Buraga

Candidate

Elena Creangă

Session: July 2017

“ALEXANDRU IOAN CUZA” UNIVERSITY, IAȘI FACULTY OF COMPUTER SCIENCE

A conceptual model of user-based motion tracking

Scientific coordinator

Conf. Dr. Sabin-Corneliu Buraga

Candidate

Elena Creangă

Session: July, 2017

DECLARATION OF ORIGINALITY AND RESPECT OF CREATIVE RIGHTS

I hereby declare that the dissertation entitled “A conceptual model of user-based motion tracking” is written by me and has never been presented to another faculty or institution of higher education in the country or abroad. I also declare that all sources used, including those taken from the Internet, are indicated in the work, in compliance with the rules of plagiarism avoidance:

• all fragments of text reproduced exactly, even in own translation from another language, are written between quotation marks and have the exact reference of the source;

• rewriting the texts written by other authors in their own words has a precise reference;

• source code, images, etc. taken from open-source projects or other sources are used with respect for copyright and have precise references;

• the summary of the ideas of other authors specifies the exact reference to the original text.

Iaşi, 03.07.2017 Elena Creangă

_____ (original signature)

DECLARATION OF CONSENT

I hereby declare that I agree that the dissertation entitled “A conceptual model of user-based motion tracking”, source code of the programs and other contents (graphics, multimedia, test data etc.) that accompany this work will be used at the Faculty of Computer Science.

I also agree that the Faculty of Computer Science, “Alexandru Ioan Cuza” University of Iași, can use, modify, reproduce and distribute for non-commercial purposes the computer programs, the executable format and the source, made by me in this dissertation thesis.

Iaşi, 03.07.2017 Elena Creangă

_____ (original signature)

Contents

Introduction

The context

As technology advances, it is important that it also continuously immerses into humans' everyday life, changing the way we interact and perceive it, up to the point that one is no longer aware of its existence. Not long ago, computers constituted the large margin of interaction with technology, from online browsing to any kind of activity performed in the digital area. However, once the smart mobile device niche developed, this monopole significantly diminished. Nowadays people are looking for ways to control their environment without being restrained to using a mouse or a keyboard and, why not, not even a smartphone.

The science of human computer interaction is the area of research and practice regarding the design, evaluation and implementation of interactive computing systems for human use and the study of major phenomena surrounding them. (Hewett et al., 2016)

In this context, new 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, to help the user interact with everyday objects.

For the purpose of this study, two such devices were chosen: Leap motion and Microsoft Kinect.

Fundamentals

Human-Computer Interaction

With naming ranging from Man-Machine Interaction to Human-Computer Interaction or Interfacing, the concept embodies the methods by which humans manipulate computers in ways that are enjoyable to use, engaging and accessible. (Fakhreddine Karray et al., 2008)

The reason for which a system is designed can be defined by the system's capabilities and the way these can help towards achieving its purpose. This translates into the main terms which are now defined under the HCI[1] umbrella: functionality and usability. (Gupta, 2012) Functionality of such a system is thus described by the set of actions and services provided to its users. However, the intrinsic value of functionality becomes visible only when the system becomes efficiently utilized by the user. The usability of a system with a specific functionality is represented by the degree by which a system can be used efficiently and properly to attain specific goals for specific users. (Fakhreddine Karray et al., 2008) Therefore, the effectiveness of a system is achieved when we strike a balance between functionality and usability. (Nielsen, 1993)

There are three levels of activities in which the user engages in when interacts with a machine: physical, cognitive and affective. While the physical aspect determines the mechanisms of interaction with the system, the cognitive one deals with the users' understanding of the system. The newer aspect of affective activities tries, besides making the experience a pleasurable one, to determine the user to continue using the application by changing attitudes and stirring up emotions. (Fakhreddine Karray et al., 2008)

The interaction of a user with a system relies on the number and diversity of the inputs and outputs used as communication channels. Each such independent single channel is called a modality. Based on the nature of different modalities, they can be classified into three categories: Visual-Based, Audio-Based and Sensor-Based. (Fakhreddine Karray et al., 2008) Following, we will detail the visual-based human-computer interaction – body tracking and gesture recognition.

[1] Human-Computer Interaction

Motion Detection

Motion detection is the primary process in the extraction of information regarding moving objects and makes use of stabilization in functional areas, such as tracking, classification, recognition, and so on. Temporal differencing, background subtraction and optical flow are among the existing techniques for moving object detection. Temporal differencing based on frame difference tries to detect moving regions by analyzing the distinct parts between two or three consecutive frames. While this technique is highly adaptive in dynamic environments, it is far from optimal when it comes to extracting complete shapes of moving objects. On the other hand, background subtraction is the most commonly used approach when it comes to still cameras. By using a statistical model of the background to which each frame gets compared, this method easily identifies objects in foreground. Optical flow represents an approximation of the local image motion and describes the proportion in which each pixel moves between sequential images, thus being by far the most successful of the three. (Nan Lu et al., 2008)

Motion Tracking

Motion Tracking is a relatively new area of the human-computer interaction that has only recently been widely accessible to everyday consumers. At a very basic level, Motion Tracking 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. Even if the terms of motion detection and motion tracking are mistakenly being swapped with one another, there is a big difference between the two. An example of motion detection is the light that turns on once it detects some movement – while it does respond to movement, it does not track it. Motion Tracking can be done in a variety of different ways, but, at its core, 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. (MathWorks, 2017) Once a particular object has been identified and its movement is tracked, the data gathered 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 an application.

Leap Motion

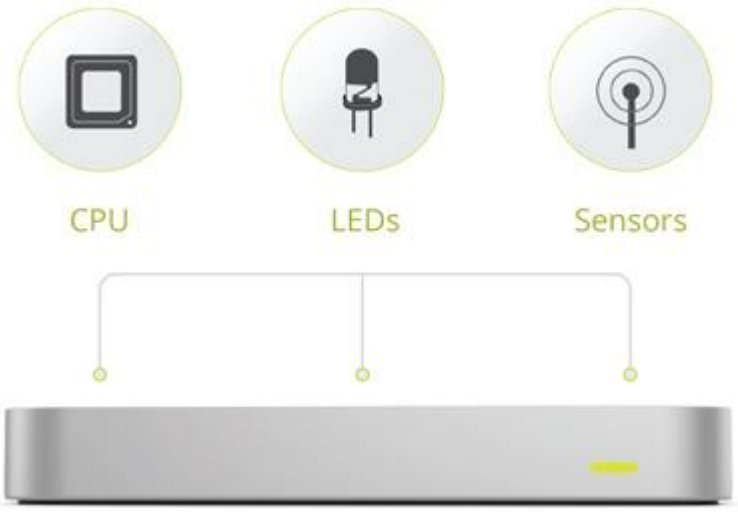

Figure 1 - Leap Motion's components

Leap Motion is a small rectangular device the size of an iPod, which can be plugged in the USB port of the device and is functional after downloading a small piece of software. From a hardware perspective, it consists of three IR[2] emitters and two IR cameras, whose main focus is recognizing and delivering positions in Cartesian space of predefined objects such as hands, fingers and finger-like tools. These positions are relative to the controller's center point, located at the position of the second, centered infrared emitter. (Frank Weichert et al., 2013) Leap Motion operates in a small proximity and with high precision, tracking frame rate and reporting discrete positions, gestures and motions. According to the specifications, it is able to detect the position of the hands and fingers with a precision of 0.01mm in a range from 25 to 600mm above the device. The software uses an internal model of a human hand to provide predictive tracking even when parts of the hand are not visible. This hand model provides positions for five fingers, but optimal tracking is achieved only when the hand's silhouette together with all its fingers are clearly visible. Even though more than two hands can appear in the hand list for a frame, it is recommended to use at most two hands for optimal motion tracking quality. (Leap Motion, 2017c)

[2] Infrared Radiation

The device can track hands, including their detailed fingers' skeleton -four bones for each one of them, except thumb-, arms and tools (that are longer, thinner and straighter than a finger). Using them, it can further map and detect gestures and motions.

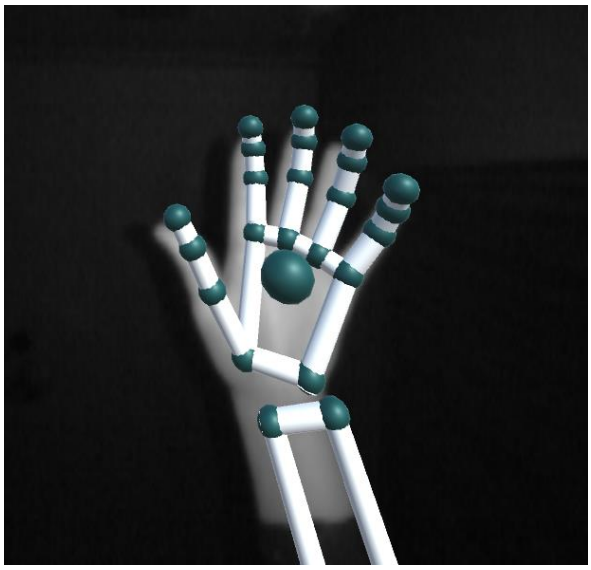

Figure 2 - Hand captured with leap motion

Pros and Cons

The device is affordable for the large public to buy and offers Software Developer Kits for programming languages like JavaScript, C#, C++, Java, Python, Objective-C and also documentation for Unity. It can be used for achieving a ubiquitous computing and ambient intelligence, to interact with technology without needing a keyboard or mouse for it. Beyond the novelty of controlling applications by detecting the movement of the fingers, it can also benefit people with physical limitations, who are unable to use a mouse. (Mitroff, 2013) Nevertheless, it can be used in a wide range of activities, from presentations to 3D[3] modeling. However, the device also has a few important low points, which might set the user off using it.

One of the main drawbacks of this device is represented by its limited sensory space and inconsistent sampling frequency which lead to a lack of accuracy. When evaluating its accuracy through the distortion of the distance between two moving points with a constant inter-point distance, the device had inconsistent performance. (Guna, 2014) If intended to be used for interaction with an application entirely

[3] Three Dimensional

based on Leap Motion, especially where control is very important, the user might face quite some difficulty getting to the wanted result or might even fail reaching it at all. The sensors do not always register hand position properly and the user often needs to repeat the movement or make extra steps to undo it if it registers the wrong way.

Another disadvantage is represented by the play-area, as everything is needed to be operated in about the same space, surrounded by the device and all the while trying not getting out of the active area.

The achievable accuracy of measuring the motion of a hand is affected by the so-called tremor, the involuntary and approximately rhythmic movement of muscles. Depending on the user's age, the tremor amplitude varies between 0.4 mm ± 0.2 mm for young individuals and 1.1mm ± 0.6mm for old individuals. The tremor, combined with the rather easily achieved tiredness of the hands - as it is not comfortable to manipulate an application with your hands focused in the same place, having no support - result in a high deviation between the desired 3D Position and the measured position. (Frank Weichert et al., 2013)

Last but not least, the inexistence of a standard set of gestures for having a consistent overall interaction with the user represents another drawback. It is therefore at the developer's latitude which gestures correspond to which actions, which understandably leads to confusing the end users when using different applications.

For the purpose of example, we mapped tracking hand gestures such as full hand rotating grip, bidirectional hand swipes and index finger pointing in a circular movement. Although the acclaimed accuracy of the device should be between 0.01mm and 0.02mm, it was never achieved in real life conditions. There were glitches registered even in normal lighting conditions and even losing track of the fingers during an activity. Even when calibrating the controller and switching between lighting settings, from room with no outside light, there has not been an improvement of any sort.

In spite of the fact that it is a revolutionary input device gesture-based human-computer interaction (Guna, 2014), Leap Motion cannot be used for complex movement patterns yet, control attempts often resulting in erratic arrow movement (Pechlaner, 2013).

Microsoft Kinect

The Kinect is a more sophisticated piece of hardware compared to the Leap Motion, being able to see, hear and understand to some extent the environment in which it is located, thanks to several different kinds of sensors and even a motor, which can change its viewing angle. The diagram below depicts its main components: an RGB[4] camera in the left corner of the Kinect, a depth sensor right next to it and a combination of an IR emitter, and an IR detector on the center of the device, that is used to detect distance from its visible field. There is also an array of microphones built into the Kinect to give it advanced directional sound detection capabilities. (Microsoft, 2017d)

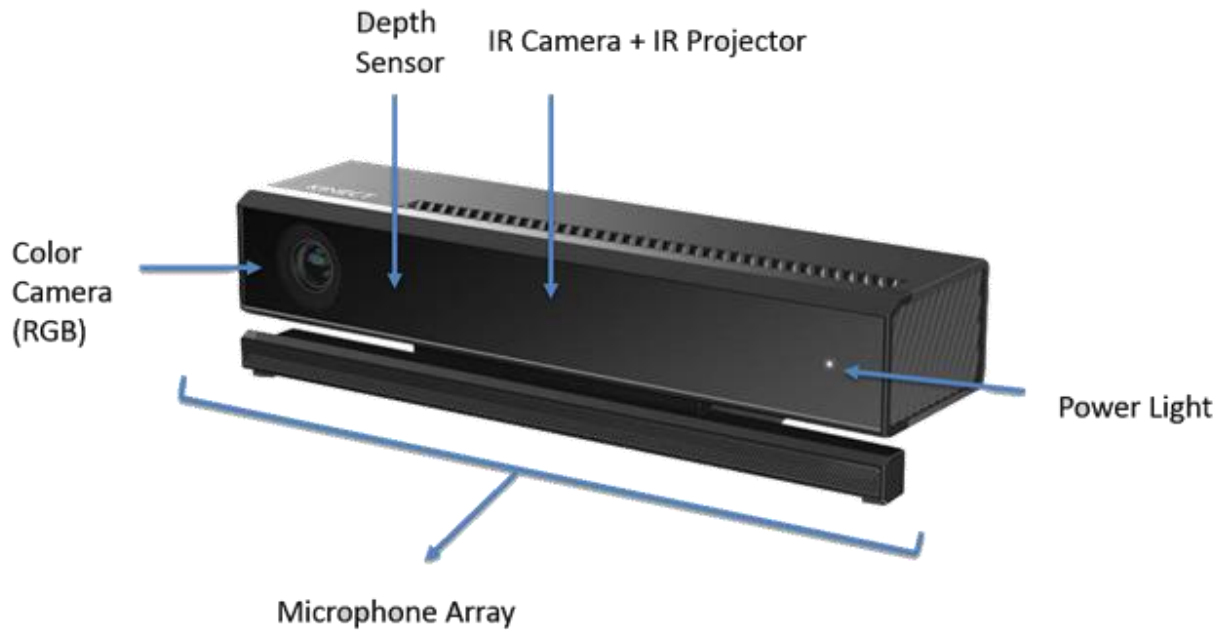

Figure 3 - Kinect sensor (Microsoft, 2015)

[4] Red Green Blue is a color model in which these colors are are added together in various ways in order to reproduce a broad array of colors.

Color Camera

The Kinect's camera is equipped with a high quality color camera which captures video at Full HD[5] resolution - 1920 × 1080 and is able to transfer the data at a rate of 30FPS[6]. It has a vertical viewing range of about 43 degrees and a slightly wider horizontal viewing range, at 57 degrees. This range is important when interacting with the Kinect, because the closer one gets to it, the more the body will be out of the range of the camera and depth sensors.

Figure 4 - Frame captured by using color camera

IR Emitter

The real power of the Kinect comes from its ability to detect depth to discern objects and movement. This power is enabled by a combination of an IR, or Infrared Emitter, and a depth sensor that is able to read the infrared light that is projected off of objects in front of the Kinect. The emitter actually emits a pseudo-

[5] High Definition
[6] Frames Per Second

random set of infrared dots. The data stream from the depth sensor maxes out at 640x480 pixels, and has the same frame rate and view angles as the RGB camera. (Microsoft, 2017d)

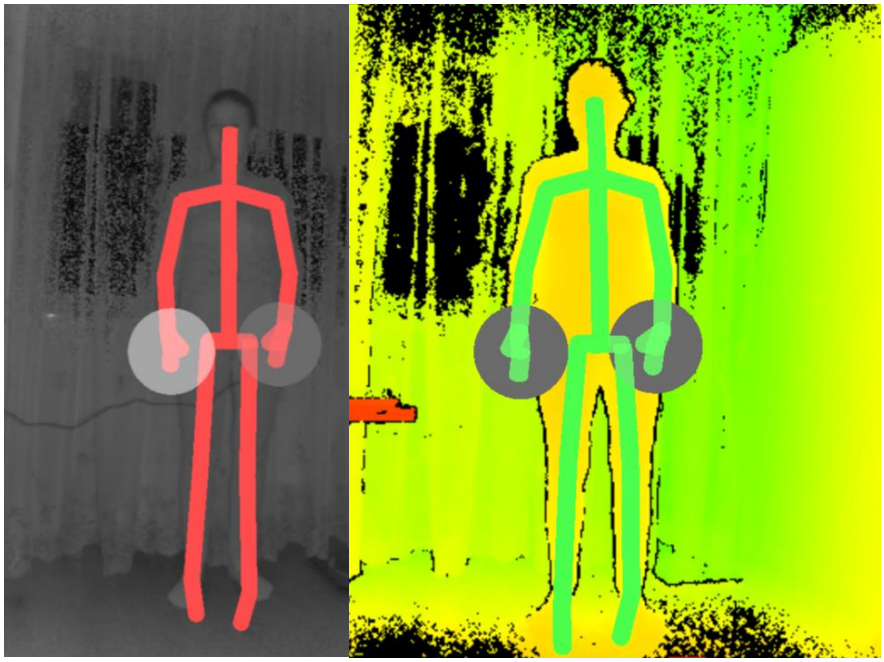

Figure 5a - Frame captured by using IR camera *Figure 5b - Frame captured by using depth camera*

With the help of both IR and depth sensor the Kinect is able to render accurate 3D visualizations of the environment captured.

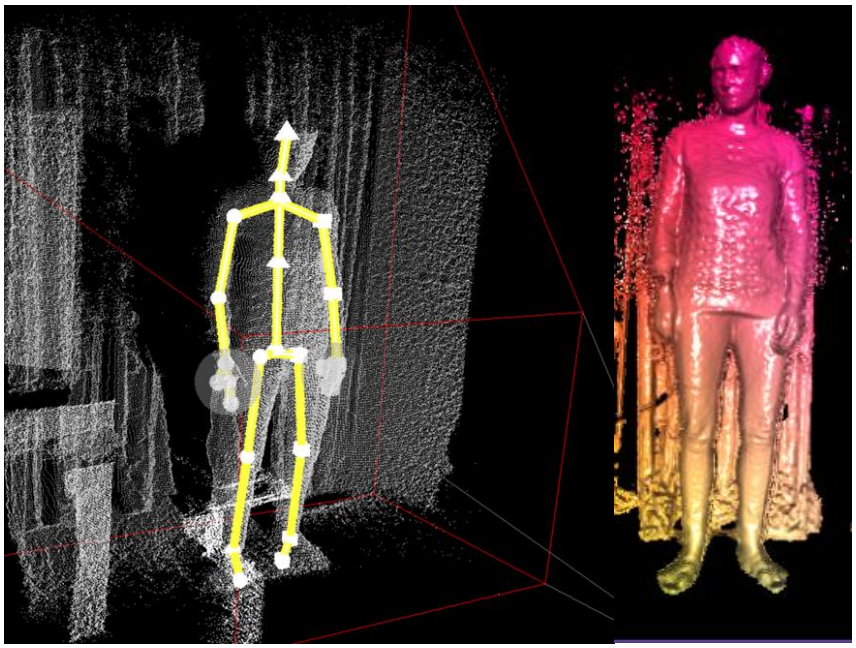

Figure 6 - Frames of 3D visualisation

Microphone Array

The Kinect has 4 built-in microphones that are used to capture sound, record audio and also find the location of the sound source and the direction of the audio wave. (Microsoft, 2017d) The microphones are laid out so that one microphone is on the left side of the Kinect and the other three are all on the right side. Just like human ears, it is able to detect where sound is coming from, by identifying the small differences in timing between when it receives the sound and from which microphones, in order to figure out where the sound is coming from. It can also use this array of microphones to do other types of processing, such as suppressing background noise and canceling echoes.

Title Motor

Besides its night vision, depth detecting, sound wave pinpointing features, the Kinect also has the capability to move itself to track the people when they go out of range of its cameras. Thus, it can tilt itself up and down in a range of 27 degrees and also has a built-in accelerometer that is able to detect if it is upright or not.

Problem Analysis

Body tracking with Kinect

Using all these sensors, the Kinect can track up to six people at the same time, constructing skeletal tracking (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 half of them (12) are analyzed. These joints include hand tips and thumbs.

The API provides access to all these sensors through its frame readers: Audio, Body, BodyIndex, Color, Depth, Infrared and LongExposureInfrared. (Microsoft, 2017b) The skeletal body tracking is obtained by using a BodyFrame item, accessed from BodyFrameReader. The BodyFrame is a sum of the 25 joints' positions, for each of the maximum of 6 persons that can be tracked simultaneously. Each joint is placed on the X Y Z axis, having a specific orientation and is assigned a specific number to identify it. The joins form the following skeleton:

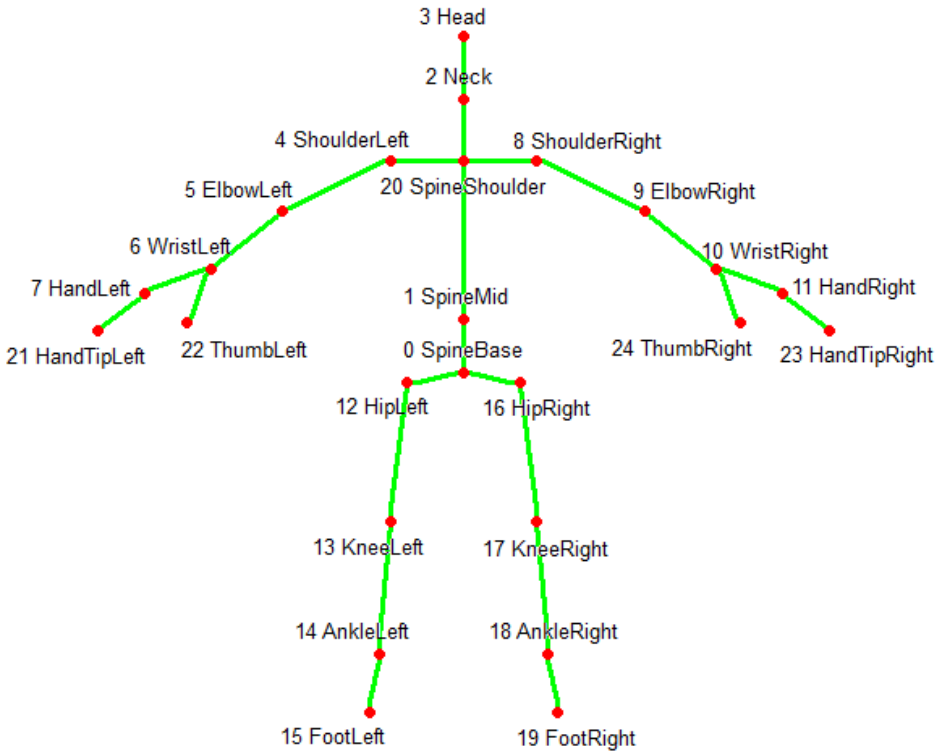

Figure 7 - Kinect body joints

Gestures

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. Waving goodbye is a gesture. Pressing a key on a keyboard is not a gesture because the motion of a finger on its way to hitting a key is neither observed nor significant. All that matters is which key was pressed.” (Kurtenbach, 1990)

Gesture recognition of humans represents the base of developing applications using Natural User Interfaces. They are used for interaction, navigation or data input. When speaking about gestures in the context of Xbox Kinect, we define them as the relative positions of some specific body joints for a given number of frames. By default, Kinect offers only 3 predefined gestures, as states for hands: Open, Closed, and Lasso.

Figure 8 – Hands' states from left to right: Open, Closed, Lasso

Gesture Segments

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. (Microsoft, 2017a) Thus, all these segments combined will then form the whole gesture. If we take the wave gesture as an example,

it happens when one raises 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).

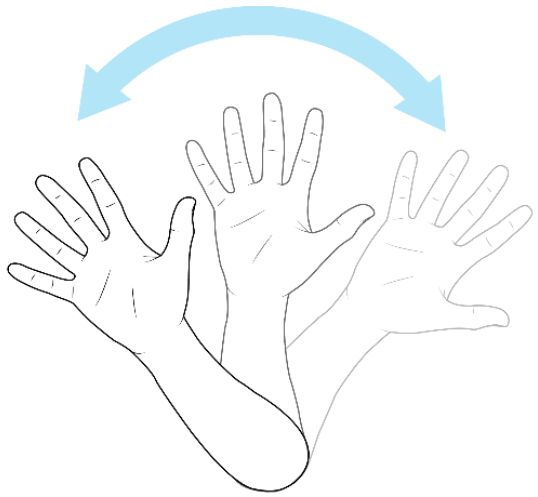

Figure 9 - Wave gesture's states (Talbot, 2014)

However, recognizing segments is not enough, if we do not take into account a degree of confidence. Let us consider the above mentioned 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. (Microsoft, 2011)

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

Analysis and Implementation

The current thesis tries to analyze and come with a conceptualization of exactly this kind, in which we can model the body and its joints' movement in a descriptive way.

Pose

The first step to creating a body is to define the joints as more than simple enumerators, but as objects which can be compared one to another using comparisons similar to natural language as follows: "Above", "Below", "After", "Before", "ToTheLeftOf", "ToTheRightOf", "AtTheSameLength", "AtTheSameHeight", "AtTheSameDepth", "OverlapsWith".

Next, joints were merged into segments, as depicted in Figure 7. This resulted in having a bone-like structure, from hands to feet which permitted identifying the pointing direction for each of them. Because there are some poses, such as both arms raised above the head – the equivalent of surrender –, we also provided the possibility to tie together mirroring segments as one entity.

In order to determine a joint's position in rapport with another, 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 ratio, we needed to calculate the shortest distance between any two given joints and add the positions of each intermediary joint to the final rapport.

For analyzing 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 pose represents the most fine-grained part of a gesture, 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

was imperious, thus providing the possibility to merge two poses (“and” operation denoted by & symbol), assume the correctness of only one (“or” denoted by | symbol) and matching the opposite of a pose (“not” denoted by !).

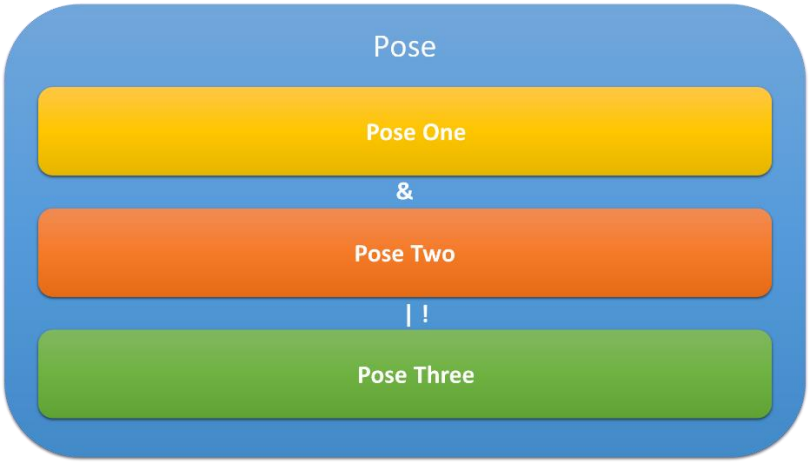

Figure 10 - Pose Model

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 Kinect frame, resulting in a degree of confidence that the two are alike. 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%.

Gestures

Just as illustrated by the guidelines of Microsoft Kinect, gestures can belong to three types: static, dynamic and continuous. (Microsoft, 2017a) Since the difference between dynamic and continuous is in the way each interacts with the application, it has not been the subject of modelling it. Instead, we focused on their general recommendation, representing gestures by a combination of segments which are executed in a specific order in a certain time frame.

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 create model more complex gestures, such as wave or clap. However, 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 executed in the exact same order. States for each gesture segment exist precisely for preventing early dismissal of a gesture as a whole. If we consider 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 10 frames 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 outcome will be Failed, reverting all previous segments' validity.

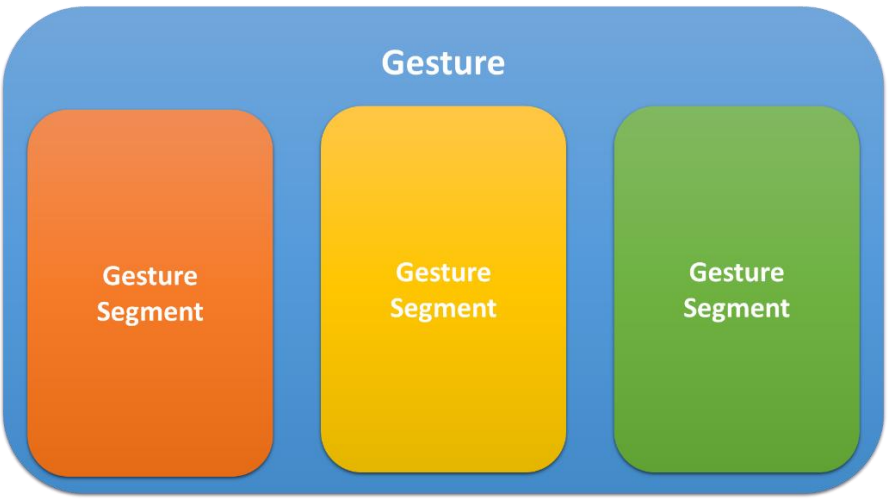

Figure 11 - Gesture Model

Ontology

Ontologies (Nicola Guarino, 2009) model and depict the relationships between concepts in a given domain. For the ease of modelling movement and interaction with Kinect sensor, we propose an ontology which represents sensor information generally, but also gesture interaction.

We defined the following classes: Sensor, User, Body, BodyPart, Part, Limb, Arm, Leg, Bone, Joint, Gesture, GestureSegment, HandState and Pose. The detailed structure is presented below:

Sensor

- represents the generic sensor which is used to identify the user;
- is able to identify users, defined by the object property “detectsUser”;
- has one individual defined: KinectSensor.

The User

- has exactly one Body, defined by the object property “hasBody”.

The Body

- is identified using a TrackingId, which is a Literal;
- has multiple BodyParts, identified using the property “hasBodyPart”.

BodyPart

- is a representation of some elements of the human body;
- is parent of Limb, Bone and Joint.

Limb

- a subclass of BodyPart;
- parent of Arm and Leg;
- has at least one Bone, defined through property “hasBone”.

Arm

- has only two disjoint individuals: RightArm and LeftArm;
- is subclass of Limb.

Leg

- has only two disjoint individuals: RightLeg and LeftLeg;
- is subclass of Limb.

Bone

- is a subclass of BodyPart;
- is formed of exactly two Joints defined through property “hasJoint” of cardinality 2;

- has 25 individuals which form the entire skeleton.

Joint

- is subclass of BodyPart;
- has 25 individuals defined.

Part

- depicts the tie between two BodyParts and an action;
- has a main BodyPart defined through the property “hasBodyPart”;
- has actions defined in relation to itself:
    - isAbove
    - isBelow
    - isAtTheSameHeightOf
    - isAtTheSameLengthOf
    - isAtTheSameDepthOf
    - isToTheRightOf
    - isToTheLeftOf
    - overlapsWith
- used to depict the relationship between different BodyParts;
- e.g. RightHand *totheLeftOf* RightElbow.

Pose

- represents a union of Parts, defined through the property “hasPart”;
- used to depict the relationship between different Parts;
- e.g. hasPart RightHandToTheLeftOfRightElbow; hasPart RightHandAboveRightElbow.

GestureSegment

- embodies a multitude of Poses defined using the “hasPose” property;
- used to depict the connection between Poses;
- e.g.
    - hasPose RightHandTopLeftOfRightElbow;
    - hasPose LeftHandBelowAtTheSameLengthWithLeftElbow.

Gesture

- illustrates a gesture
- has at least one GestureSegment defined, using “hasGestureSegment” property;
- e.g. of individual: WaveRight.

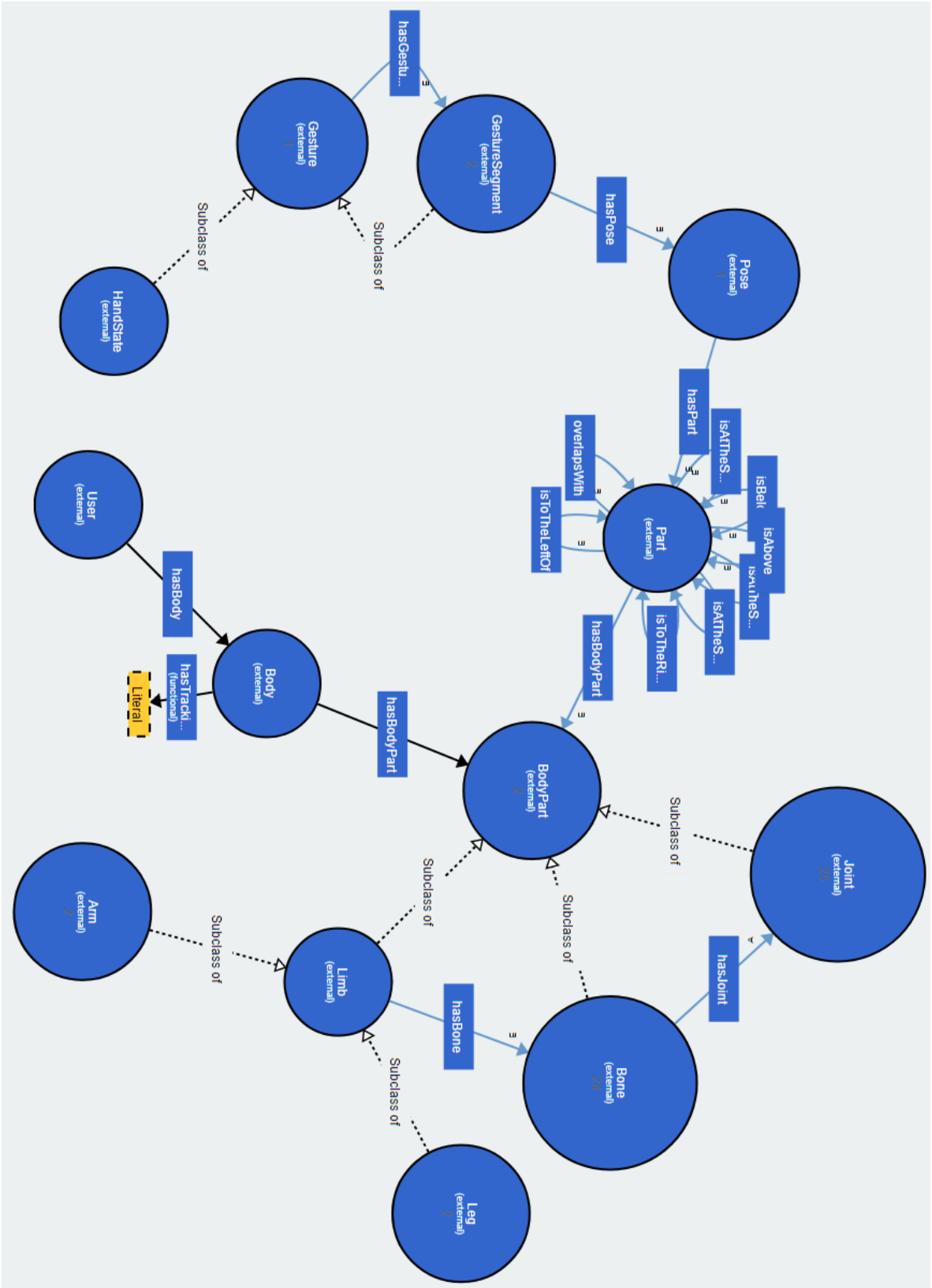

Figure 12 - Visualisation of ontology using WebVOWL[i]

Use cases

In order to demonstrate the potential of the concept presented, some use cases are necessary to be analyzed. For this purpose, we chose to develop two projects: one about manipulation of shapes in 3D space and one about usability testing.

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 our playground. Initially the board presents itself filled with three shapes, rendered using OpenGL: a sphere, a cube and a pyramid. Since the main manipulators of our shapes are the hands, we want to keep track of every move they make. Therefore, we also represent them on the board, as spheres. For better visualizing their position on the x-y-z axis, 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.

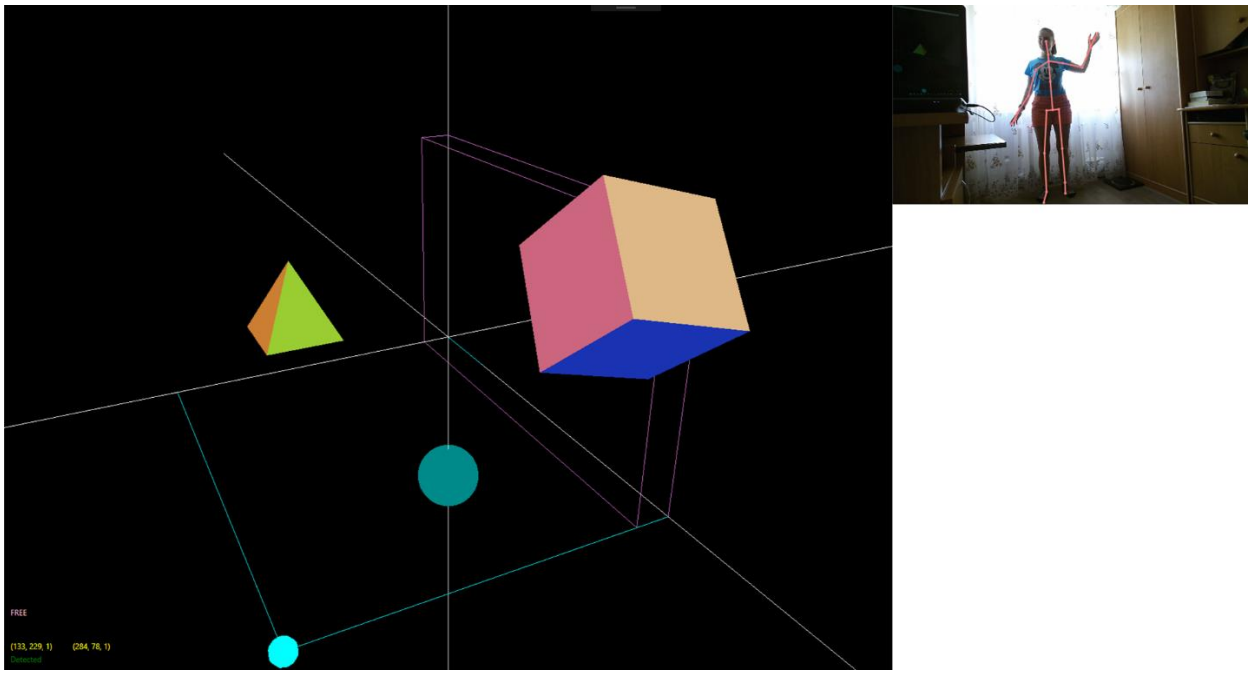

Figure 13 - Image of the shape module

The mapped behavior of the drawing board is represented by both static, and dynamic gestures.

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. 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 associated with “x” symbol which is associated with closing or deleting a program or activity and thus, when detected, the drawing board erases all its shapes.

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% standard, 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, whereas successfully completing the two-way movement, three times in a row, will result in successfully recognizing it. The waving gesture stands both for salutation and for erasing a drawing board. Combining the latter with the already familiar shake necessary for undoing written text on the iPhone, we mapped this gesture to undoing the previous movement on the board.

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 or game (e.g. Kinect Sports, Dance Central), therefore, when detecting it, the board will reset to its initial state.

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, one 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.

This playground for 3D shapes represented the first proof for the ease of modeling gestures using this conceptualization.

Usability testing

Movement tracking and gesture detection are used in all applications which are destined for Kinect sensor and the model we created can be extrapolated and used for developing any of them. Currently, every application has its own – better or worse- way 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. (Jarrett Webb, 2012)

This is why for the second use case 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. We chose as samples the screens of two applications: a dressing room (Fitnect[7]) and a math game (Jumpido[8]).

[7] http://www.fitnect.hu

[8] http://www.jumpido.com

Fitnect

Fitnect is a virtual store which uses augmented reality to create a 3D fitting room system. 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 gestures used should not be too complex.

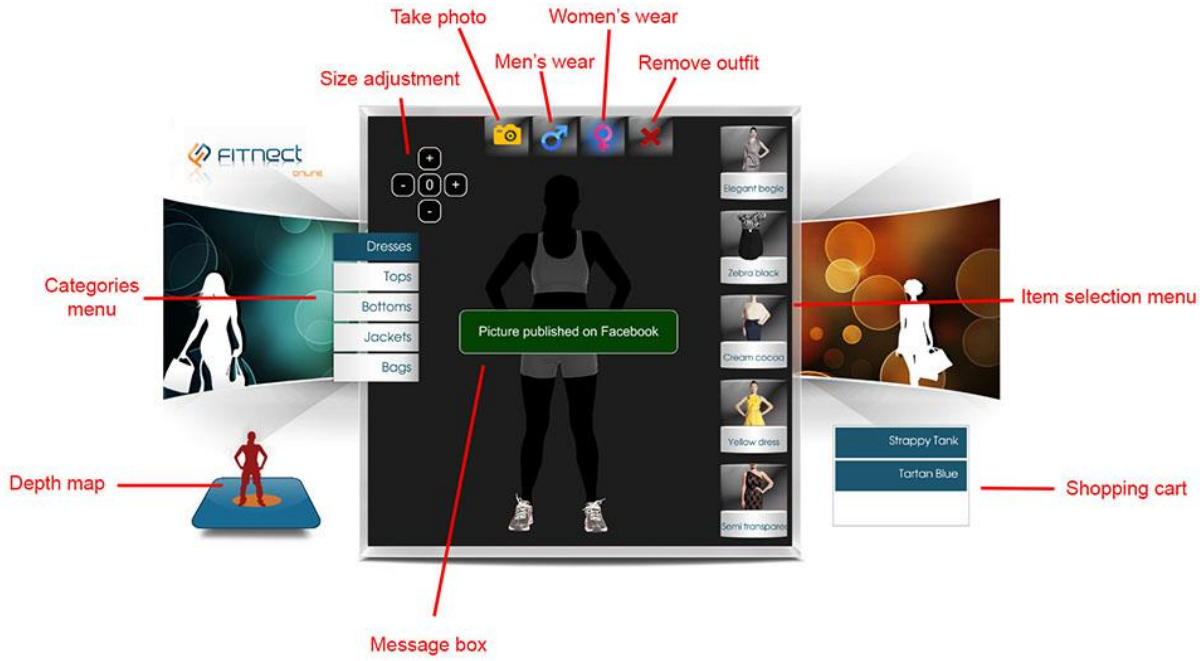

Figure 14 - Fitting Room screen from fitnect wiki[9]

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.

[9] http://wiki.fitnect.hu/index.php/Main_Page

Figure 15 - Surrender pose from Fitnect's wiki

The mapping of this pose was created 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^{o} 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.

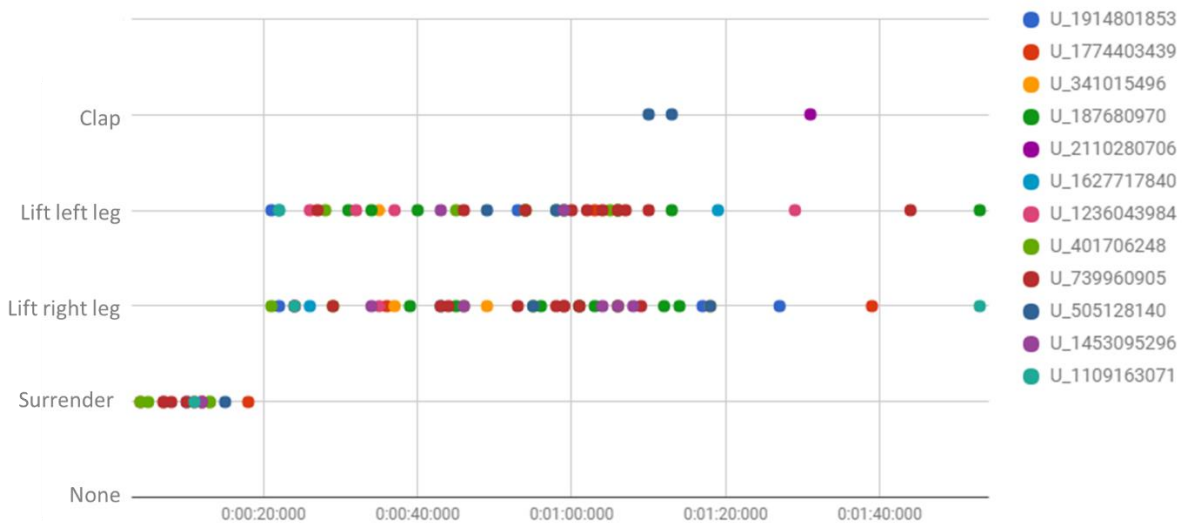

Figure 16 - Gestures captured while using Fitnect

As we can see in the graphic above, 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 users. 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 irresponsiveness of the application. The average interaction of one user lasted one minute and 30 seconds.

Jumpido

Designed having in mind children as target audience, Jumpido 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.

Figure 17 - Children interacting with Jumpido (Jumpido, 2017)

We chose to analyze a game with a football theme. 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.

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.

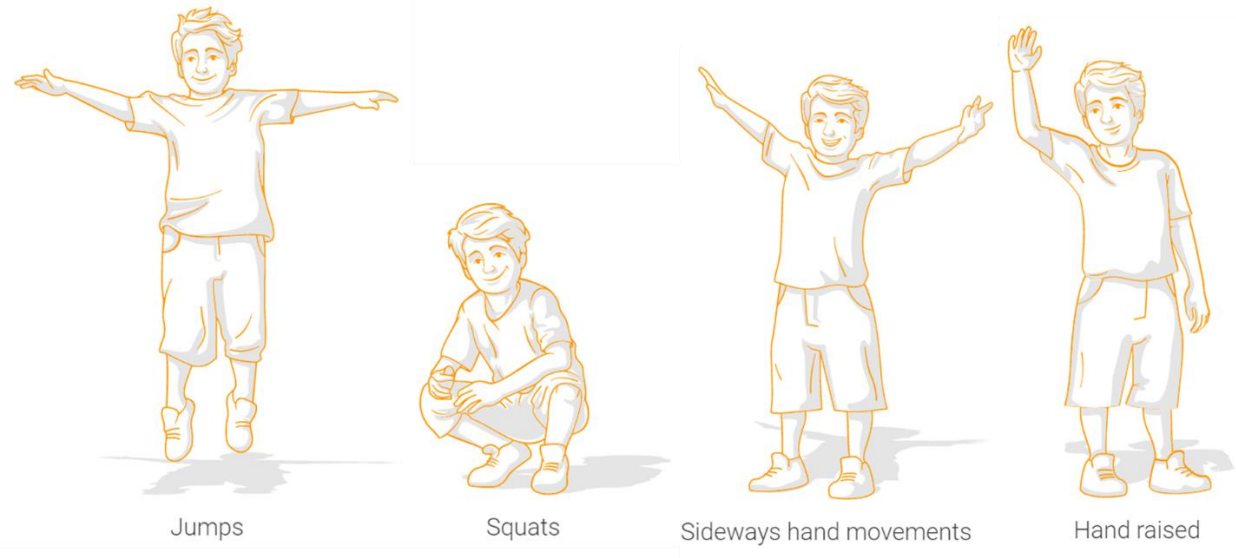

Figure 18 - Gestures used by Jumpido (Jumpido, 2017)

While detecting a waving hand is rather straight forward, 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 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 modelling 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.

Figure 19 - Gestures captured while using Jumpido

The graphic illustrated below depicts the gestures performed by users while playing a module of Jumpido. 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 notice that the users figure what they have to do from as early as the first 30 seconds of interaction. By looking at the graphic we can also deduce that the majority of the 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 gestures detected by a total of 8 users, 28% were jumps, 37% were squats, 23% were kicks with the right foot and 12% were kicks with the left foot.

Conclusions

Motion tracking and gesture recognition techniques allow us depict new ways of interacting with computers, blurring the line between imagination and reality. Devices such as Leap Motion and Microsoft Kinect open new possibilities in the realm of ubiquitous computing, by allowing ease of access of applications through posture analysis.

The current thesis illustrates an analysis of the current state of art with regard to Motion Tracking and proposes a conceptual model for manipulation of gesture based applications. From there on, we modelled a way of identifying postures and gestures in an easier way, using a more natural language instead, having relative comparisons between the two. Afterwards, we tested the accuracy of our concept by creating a library which we then used for creating a playground for manipulating 3D shapes and a usability testing tool for analyzing existing applications.

While the conceptual model we created for modelling postures and gestures is designed having Microsoft Kinect as its main focus, it can also be extended to other tracking devices.

An important future aim is represented by the adaptation of the model to be able to analyze facial expressions in order to determine unobtrusive psychological data such as head pose and eye tracking. As each gesture can have a different meaning depending on the context, the mimic of the face would represent an improvement in correctly detecting the user's moods. Nevertheless, this development would facilitate the identification of weaknesses and performances for users' activities, allowing to create segment alteration instructions for different levels of complexity in the applications, in order to increase their engagement.

References

Buraga Sabin-Corneliu Human-Computer Interaction [Online]. - 2017. - https://profs.info.uaic.ro/~busaco/teach/courses/hci/hci-film.html.

Buraga Sabin-Corneliu Web Application Development [Online]. - 2017. - https://profs.info.uaic.ro/~busaco/teach/courses/wade/web-film.html.

D. Allemang J. Hendler Semantic Web for the Working Ontologist (2nd Edition) [Book]. - [s.l.] : Morgan Kaufmann, 2011.

Fakhreddine Karray et al. Human-Computer Interaction: Overview on State of the Art [Journal] // International Journal On Smart Sensing And Intelligent Systems. - [s.l.] : Massey University, March 2008. - 1 : Vol. 1. - 11785608.

Frank Weichert et al. Analysis of the Accuracy and Robustness of the Leap Motion Controller [Journal]. - Basel, Switzerland : MDPI AG, May 2013. - 5 : Vol. 13. - pp. 6380-6393. - 1424-8220.

Guna Jože An Analysis of the Precision and Reliability of the Leap Motion Sensor and Its Suitability for Static and Dynamic Tracking [Journal]. - Basel, Switzerland : MDPI AG, 2014. - 2 : Vol. 14. - pp. 3702-3720. - 1424-8220.

Gupta Rachit Human Computer Interaction – A Modern Overview [Journal] // Int.J.Computer Technology & Applications. - September 2012. - 5 : Vol. 3. - pp. 1736-1740. - 2229-6093.

Hewett et al. ACM SIGCHI Curricula for Human-Computer Interaction [Online] // hcibib.org. - 2016. - http://hcibib.org/.

Jarrett Webb James Ashley Beginning Kinect Programming with the Microsoft Kinect SDK [Book]. - New York : APress, 2012.

Jumpido Innovative school in Sofia gives Jumpido flying colors [Online] // Jumpido. - 2017. - http://www.jumpido.com/en/education/kinect/school/case-studies/sofia137.

Kurtenbach Gordon and Eric A. Hulteen Gestures in Human-Computer Communication [Book]. - [s.l.] : Addison-Wesley Publishing Company, 1990.

Leap Motion Leap API Overview [Online] // Leap Motion. - 2017c. - https://developer.leapmotion.com/documentation/csharp/devguide/Leap_Overview.html.

MathWorks Motion-Based Multiple Object Tracking [Online] // MathWorks. - 2017. - https://www.mathworks.com/help/vision/examples/motion-based-multiple-object-tracking.html.

Microsoft Human Interface Guidelines [Online] // Microsoft.com. - 2017a. - https://msdn.microsoft.com/en-us/library/jj663791.aspx.

Microsoft Kinect for Windows v2 - The next generation of PC applications [Online]. - 2015. - http://blogs.microsoft.co.il.

Microsoft Kinect Frame Source Types [Online] // MSDN. - 2017b. - https://msdn.microsoft.com/en-us/library/windowspreview.kinect.framesourcetypes.aspx.

Microsoft Kinect Hardware [Online] // Microsoft. - 2017d. - https://developer.microsoft.com/en-us/windows/kinect/hardware.

Microsoft Writing a gesture service with Kinect for Windows SDK [Online] // MCS UK Solution Development Team. - 2011. - https://blogs.msdn.microsoft.com/mcsuksoldev/2011/08/08/writing-a-gesture-service-with-the-kinect-for-windows-sdk/.

Mitroff Sarah Not magic, motion control: The Leap controls your computer with a flick of a finger [Online] // VentureBeat. - 05 21, 2013. - https://venturebeat.com/2012/05/21/the-leap-motion-control/.

Nan Lu et al. An Improved Motion Detection Method for Real-Time Surveillance [Journal] // IAENG International Journal of Computer Science. - 2008.

Nicola Guarino Daniel Oberle, Steffen Staab What Is an Ontology? [Book Section] // Handbook on Ontologies / book auth. Steffen Staab Rudi Studer. - Berlin Heidelberg : Springer-Verlag, 2009.

Nielsen Jakob Usability Engineering [Book]. - San Francisco : Morgan Kaufmann, 1993.

Pechlaner Bernhard Review Leap Motion Motion Control Technology [Online] // NotebookCheck. - 09 08, 2013. - https://www.notebookcheck.net/Review-Leap-Motion-Motion-Control-Technology.98821.0.html.

Semantic Web Semantic Web [Online] // SemanticWeb.org. - 2017. - http://semanticweb.org/wiki/Semantic_Web.html.

Talbot Paige Robotic-Interaction Hand Gesture Illustrations [Online] // Simple-ism. - 2014. - http://simple-ism.net/robotic-interaction-hand-gesture-illustrations/.

Theodosis Georgiou Yiannis Demiris Adaptive user modelling in car racing games using behavioural and physiological data [Journal] // User Modeling and User-Adapted Interaction. - [s.l.] : Springer, June 2017. - 2 : Vol. 27. - pp. 267-311.

Tim Berners-Lee James Hendler and Ora Lassila The Semantic Web [Journal] // Scientific American. - 06 17, 2001.

[i] http://visualdataweb.de/webvowl/