Tools for Collecting Users' Touch and Free-Hand Gestures on Mobile Devices in Experimental Settings

Bogdan Florin GHERAN, Gabriel CRAMARIUC, Ionela RUSU
MintViz Lab | MANSiD Research Center
University Stefan cel Mare of Suceava
Suceava 720229, Romania
bogdan.gheran@gmail.com, gabi.cramariuc@gmail.com, ionelar@eed.usv.ro

Elena-Gina CRĂCIUN
University Stefan cel Mare of Suceava
Suceava 720229, Romania
ginac@eed.usv.ro

***Abstract*— Gesture interaction is now available on a variety of devices that expose touch and/or free-hand gesture interfaces. However, users' capacity to interact efficiently with computing systems using gestures is still to be understood. To this end, the community needs appropriate software tools to record users' gesture input in both experimental and live settings. In this work, we present three software tools that capture and record users' touch and free-hand gesture input on mobile devices. These tools make together a general and unique system that can collect precise data about users' free-hand and touch gestures. The variety of gesture data and gesture measurements reported by our tools will enable practitioners and researchers to form a thorough understanding about their users' performance during gesture interaction with mobile devices. Our tools are free to download and use to collect users' gesture performance on mobile devices in various experimental settings.**

Keywords*— gesture input, touch gestures, free-hand gestures, finger gestures, Leap Motion, Ring Zero, mobile devices, software tool, gesture acquisition, experiments, gesture measurements

I. INTRODUCTION

Gesture-based interfaces have great potential for intuitive and natural interaction with computing systems. Today's commercial devices allow human gestures to be captured in various forms, such as touch input, free-hand, or whole-body movements. However, users' capacity to interact efficiently using gesture commands with computing devices still needs to be understood. For instance, studies on touch-screen technology have shown that users prefer gesture commands that are different from those proposed by expert designers [13,14]. Touch interaction platforms are designed for quick time response to gestures, while this puts pressure on software developers to create suitable processing algorithms. In terms of hand and body movements, sophisticated motions performed on todays' smartphones can be acquired using accelerometers, gyroscopes, and magnetometers, which enable practitioners to leverage device motion to infer users' gestures and actions [1,4,8,11,16,18].

Despite researchers' efforts to recognize the human gesture input on mobile devices and to understand how users actually perform gestures [10-16,19,22-23], we believe that the Human-Computer Interaction community still needs appropriate software tools to record users' gesture input in experimental settings. In this paper, we present several such tools that capture and record users' touch and free-hand gesture input on mobile devices: *TapExperiment* is a tool running on Android systems that captures users' touch performance and characterizes it with various measures, such as target offset, tap time, pressure, or touch area. *FingerGestureExperiment* is a tool that collects free-hand finger gestures from a wearable device and reports finger orientation parameters using the Ring Zero device [5]; and *LeapGestureExperiment* is a tool that captures free-hand 10-finger gesture input in mid-air using the Leap Motion controller [6]. The novelty of these tools is that they can be used together to collect simultaneously different hand gesture data, both on touch and in mid-air. Another important feature is that they offer the user a complete and unique dataset with hand gesture descriptions, composed by various touch and mid-air gesture measures. This information can further be used as experimental data for different gesture recognition algorithms, for testing other gesture user interfaces or for constructing 3D hand models.

In this work, we present the functionality of these tools, the data they record and the gesture measurements they report. Our software tools are free to download and use by contacting the first author to allow researchers and practitioners to collect users' gestures on mobile devices in experimental settings.

II. RELATED WORK

The development of different hand gesture interfaces has been reflected in the proposals of various tools for gesture acquisition, recognition, and/or analysis. For instance, Long *et al.* [2] introduced an interactive system for designing pen gestures (quill) that assists designers in creating and improving the quality of gesture commands for pen-based interfaces, both in terms of how a computer is able to recognize those gestures and in terms of the likelihood of gesture confusion among people. Long *et al.* found that users do not understand how gesture recognizers work and, therefore, finding and fixing problems with gesture recognition of custom user-defined gesture commands was difficult for users. Inspired by Long *et al.*, Ashbrook *et al.* [1] proposed MAGIC, a gesture design

978-1-5090-1993-9/16/$31.00 ©2016 IEEE

toolkit that allows practitioners to design, evaluate, and test motion gestures. MAGIC allows designers to test gesture input generated in daily activity, while it ignores unintended gesture input and, consequently, increases the recognition rate of the system. Vatavu *et al.* [4] proposed a new approach of presenting gesture recognition for control applications in a service-oriented event-driven manner: Gesture Profile for Web Services (GPWS), which represents a service-oriented architecture (SOA) for smart environments. Chera *et al.* [17] proposed gesture ontologies for informing service-oriented architectures for gesture processing and recognition.

Other tools exist to analyze gestures produced by users, such as GREAT (the Gesture Relative Accuracy Toolkit) [10], GECKo (Gesture Clustering Toolkit) [12], GHOST (Gesture Heatmaps Toolkit) [9] or GDA-TK (the Gesture Dimensionality Analysis Toolkit) [11]. While these tools are useful to compute gesture measures, to visualize how these measures vary within and between users, and to optimize gesture representation in terms of gesture dimensionality and gesture cardinality, they need a gesture set to exist a priori. The contribution of this work is to present the community with software tools to acquire gesture input from touch-screen and free-hand gesture input devices in experimental settings. We describe our software tools in the next sections.

III. Tool for Collecting Touch Gestures

The *TapExperiment* application was developed to collect touch gestures on mobile devices. Users must acquire targets represented as circles as fast and accurately as possible. Targets periodically shift position and size. To prevent users from touching accidentally other areas of the screen, touch targets can be designed to be sufficiently large and far apart from each other. (The average adult finger pad size is about 10*mm* wide.) We took into consideration recommendations from Google Android SDK according to which the minimum touch target size of any element should be 48 × 48 *dp* [20]. Google Android operating system runs on a variety of devices with different screen sizes and pixel densities. Although our tool scales and resizes automatically to work on different screens, we optimized the code and converted the targets physical dimensions to the device *dpi*. Our tool was therefore designed to be device independent. The zone with the highest visibility is the central area, but complex touch experiments look at how various categories of users (*e.g.*, with visual impairments) can interact with targets placed near the margins of the device and in corners. Therefore, the display of targets in our tool focused on these areas, while only approximately 10% of all targets are located in the center.

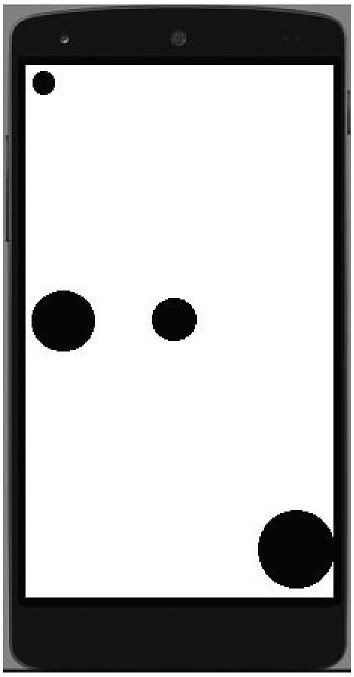

Fig. 1 *TapExperiment* application showing different touch targets.

Once the application is launched, it will start to display targets one by one. The user must touch each target before proceeding to the next one. If the touch point from the moment the user lifts the finger off of the touchscreen is at a certain distance (offset) from the center of the target, we consider that the trial was successful and proceed to the next target. In case of failure, the user receives a warning sound and has two more tries to accomplish the same trial. If these two tries are also registered as failures, the application proceeds to the next trial. The modality to display is consistent for all targets; it uses a high contrast for optimum visibility: white background and black targets. The application runs in full screen mode with the upper area of the display, usually used by the Android system to display notifications, being suppressed not to be touched by mistake. Also, once the application is launched it is locked on the screen and therefore by accidentally pushing hard or soft keys it is not closed or sent into the background and therefore measured parameters are not altered. Using smart device sensors, we also monitor the movement of the device by capturing device acceleration and orientation. It is known that the sensors generate noise which affects measurements, so we applied a filter:

$$\begin{gathered}
G_X = \alpha \cdot G_X + (1-\alpha) \cdot event.g_X \\
G_Y = \alpha \cdot G_Y + (1-\alpha) \cdot event.g_Y \\
G_Z = \alpha \cdot G_Z + (1-\alpha) \cdot event.g_Z \\
g_X = event.g_X - G_X \\
g_Y = event.g_Y - G_Y \\
g_Z = event.g_Z - G_Z
\end{gathered} \tag{1}$$

where g_X, g_Y and g_Z are the values we report and $event.g_X$, $event.g_Y$, $event.g_Z$ are the acceleration data reported by the device. Linear acceleration does not include gravity. Gestures data is stored into XML files, one file per user. Figure 2 shows an example of an XML data file. The following measures are computed by our application:

- Trial Time - the moment when the trial starts and the target is first displayed on the screen.
- Target x, y, r, and tolerance - coordinates of the displayed target, the circle's radius, and the offset.
- DeviceAcc X, Y, Z - the device records the medium acceleration on all three dimensions from the moment the target is displayed until the touchscreen is touched.
- Touch X, Y – coordinates of the touch event. All the points are recorded, even if the user moves his finger on the touch surface when accomplishing the task.

```
<Trial time="15:02:26:400">
    <Target Tolerance="26" R="26" X="1241" Y="370"/>
    <DeviceAcc X="0,3024" Y="0,3441" Z="0,1984"/>
    <Touch X="1245" Y="368" T="852250" Minor="1,2695" Major="1,5234" Orientation="1,5708" Area="0,0216"
        Pressure="0,0941" ID="0"/>
    <Touch X="1240" Y="368" T="852287" Minor="0,2539" Major="0,2539" Orientation="1,5708" Area="0,0039"
       Pressure="0,1098" ID="0"/>
    <Touch X="1239" Y="368" T="852292" Minor="0,2539" Major="0,2539" Orientation="1,5708" Area="0,0039"
       Pressure="0,1098" ID="0"/>
    <Touch X="1238" Y="371" T="852300" Minor="0,2539" Major="0,2539" Orientation="1,5708" Area="0,0039"
       Pressure="0,0902" ID="0"/>
    <Touch X="1237" Y="372" T="852306" Minor="0,2539" Major="0,2539" Orientation="1,5708" Area="0,0039"
       Pressure="0,0902" ID="0"/>
    <Touch X="1236" Y="374" T="852314" Minor="0,2539" Major="0,2539" Orientation="1,5708" Area="0,0039"
       Pressure="0,0392" ID="0"/>
    <Touch X="1235" Y="375" T="852322" Minor="0,2539" Major="0,2539" Orientation="1,5708" Area="0,0039"
       Pressure="0,0392" ID="0"/>
    <Result Success="OK" Offset="7,8102"/>
    <Time TaskTime="907" TouchTime="90"/>
    <DeviceTilt X="91.0" Y="124.0" Z="34.0"/>
</Trial>
```


Fig 2. Data example output from the *TapExperiment* software application (currently at version 1.0) using the XML format.

- AREA, MAJOR, MINOR - size of the touched area. When a finger touches the screen, it may be represented as an ellipse for which we record its geometric properties
- ORIENTATION – Returns the orientation of the device.
- T – the elapsed time for current event in milliseconds.
- PRESSURE – records the pressure push of the finger on the touch-screen. This parameter varies between 0 and 1, but the values depends on device manufacturer.
- ID – In case a user touches the screen with multiple fingers simultaneously, all this data is recorded.
- RESULT SUCCESS - with two values OK and NO, depending on the value of Result Offset. The trial is saved regardless whether the target was touched or not; offset to determine whether the trial was successfully accomplished, we calculate the distance between the touch point recorded at the Touch-Up event and the center of the target. This value is compared to the Tolerance threshold.
- TASKTIME – the elapsed time (in milliseconds) between the targets' display time to the moment the task is accomplished.
- TOUCHTIME - the timestamp (in milliseconds) when the touch-screen of the device is being touched.
- DEVICETILT X, Y, Z - the device orientation on 3-axis obtained at the moment the user touches the screen.

IV. TOOL FOR COLLECTING FINGER GESTURES

FingerGestureExperiment is an Android application that was developed to collect finger gesture data during interaction with smart devices. This application uses the Ring Zero device [5] that incorporates sensors which detect finger orientation in time and, consequently, finger movement. The communication between the Ring Zero and the smart device is implemented using Bluetooth. After pairing the ring with a touch-screen device, we collect data at its standard delivery rate (currently 3 measurements per second, as provided by the Ring Zero public beta test SDK [21]). Data from the Ring Zero device is represented using quaternions, which represent a number system that extends complex numbers primarily used in computer graphics to represent the orientation and rotation of an object in three dimensions. A quaternion is a four-tuple formed by a scalar and a vector $q[w,v]$ where w is a scalar and v a vector in 3D:

$$q = \left[w + x \cdot i + y \cdot j + z \cdot k \right] \tag{2}$$

where w, x, y and z are all scalars and

$$i^2 = j^2 = k^2 = -1 \tag{3}$$

Quaternions can be used to compute Euler's angles that describe the rotation of the Ring Zero device in 3-D space:

$$\Phi = \arctan \frac{2 \cdot (w \cdot x + y \cdot z)}{1 - 2 \cdot (x^2 + y^2)} \tag{4}$$

$$\theta = \arcsin(2 \cdot (w \cdot y - z \cdot x)) \tag{5}$$

$$\Psi = \arctan \frac{2 \cdot (w \cdot z + x \cdot y)}{1 - 2 \cdot (y^2 + z^2)} \tag{6}$$

Φ (roll) is the rotation around the X-axis, θ (pitch) is the rotation around the Y-axis, and Ψ (yaw) represents the rotation around the Z-axis like in next figure:

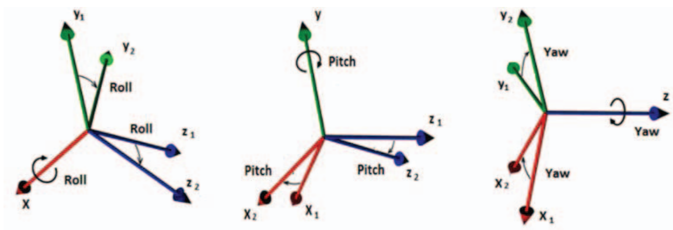

Fig 3. Diagrams showing Euler's angles reported on a 3-axis

However, this standard approach has some problems. Because trigonometrical functions *arctan* and *arcsin* return values between $-\frac{\pi}{2}\ and\ \frac{\pi}{2}$ we can't have all possible orientations. To fix this problem, we need to replace those functions with new formulas:

$$\Phi = \text{atan2}\cdot[2\cdot(w\cdot x + y\cdot z), 1-2\cdot(x^2+y^2)] \quad (7)$$

$$\theta = \text{asin}[2\cdot(\text{w}\cdot\text{y-z}\cdot\text{x})] \quad (8)$$

$$\Psi = \text{atan2}\cdot[2\cdot(\text{w}\cdot\text{z+x}\cdot\text{y}),\ 1\text{-}2\cdot(\text{y}^2\text{+z}^2)] \quad (9)$$

The *FingerGestureExperiment* was developed using the Ring Zero public beta test SDK [21]. Events that can be processed are the following: receive device orientation in quaternion format; receive gesture notification; receive tap and long tap events. To cycle between these events, a control mode was implemented. For our experiment of collecting users' finger gestures, we extract the w, x, y, z values. The advantages of using quaternions consist in that they offer fast multiplication as they represent pure rotation. One example of data collected using our application is presented in Figure 5.

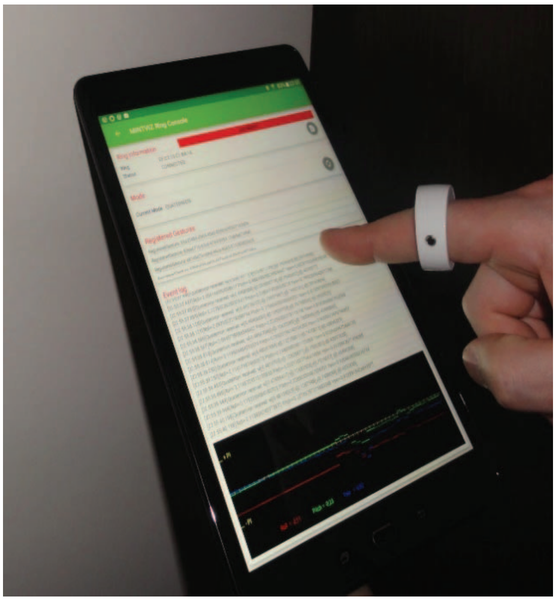

Fig. 4 Snapshot of the *FingerGestureExperiment* application (currently at version 1.0) running while collecting data from the Ring Zero device.


```
<Timestamp = 15:17:28:991 Quaternion: w=[-0.83887476] x=[-0.014524532] y=[-0.0025557736] z=[-0.54412466]>
<Timestamp = 15:17:28:991 Roll = 0.027154983490730773 Pitch = -0.011518618551500973 Yaw = 1.1506755995066125>
<Timestamp = 15:17:29:382 Quaternion: w= [-0.83882004] x=[-0.017179817] y=[-8.603176E-4] z=[-0.544137]>
<Timestamp = 15:17:29:382 Roll = 0.0297666357950758 Pitch = -0.017253899189742714 Yaw = 1.1506554988125188>
<Timestamp = 15:17:29:674 Quaternion: w=[-0.8388086] x=[-0.01676308] y=[-0.0011328785] z=[-0.5441671]>
<Timestamp = 15:17:29:674 Roll = 0.029363120868641873 Pitch = -0.016344023219710038 Yaw =1.1507352367422088>
<Timestamp = 15:17:30:064 Quaternion: w=[-0.8388215] x=[-0.0160901] y=[-0.0014600987] z=[-0.5441668]>
<Timestamp = 15:17:30:064 Roll = 0.028589655313210813 Pitch = -0.015062443490329857 Yaw = 1.1507454058337436>
<Timestamp = 15:17:30:358 Quaternion: w=[-0.83885705] x=[-0.015793804] y=[-0.0016726824] z=[-0.54412]>
<Timestamp = 15:17:30:358 Roll = 0.028324484353462 Pitch = -0.014381663238996934 Yaw = 1.1506397280253469>
<Timestamp = 15:17:30:746 Quaternion: w=[-0.8388401] x=[-0.015565095] y=[-0.0018249061] z=[-0.5441522]>
<Timestamp = 15:17:30:746 Roll = 0.02810571041644209 Pitch = -0.013878396123806046 Yaw = 1.1507208577536692>
<Timestamp = 15:17:31:096 Quaternion: w=[-0.8062889] x=[0.064037904] y=[0.31170258] z=[-0.49863702]>
<Timestamp = 15:17:31:096 Roll = -0.47895525481506745 Pitch = -0.45424201701174216 Yaw =1.2204653731868853>
<Timestamp = 15:17:31:450 Quaternion: w=[-0.7297072] x=[0.32128114] y=[0.38142183] z=[-0.46778545]>
<Timestamp = 15:17:31:450 Roll = -1.0240307616162059 Pitch = -0.2589556990351296 Yaw = 1.286221863958013>
<Timestamp = 15:17:31:782 Quaternion: w=[-0.7955554] x=[0.2578619] y=[0.45126063] z=[-0.31138837]>
<Timestamp = 15:17:31:782 Roll = -0.983935021366348 Pitch = -0.5912692512339381 Yaw = 1.0697433879527898>
<Timestamp = 15:17:32:112 Quaternion: w=[-0.6276754] x=[0.61872965] y=[0.46788365] z=[-0.065437265]>
<Timestamp = 15:17:32:112 Roll = -1.80901762260143 Pitch = -0.5309840798517824 Yaw = 0.8736874207778267>
<Timestamp = 15:17:32:451 Quaternion: w=[-0.44260022] x=[0.75050396] y=[0.47001496] z=[0.1411906]>
<Timestamp = 15:17:32:451 Roll = -2.3895627268412123 Pitch = -0.6789621173500108 Yaw = 0.841954944686083>
```


Fig 5. Data example output from the *FingerGestureExperiment* application (currently at version 1.0) using the XML format.

V. Tool for Collecting Free-Hand Gestures

In order to collect users' free-hand gestures input for experimental scenarios, we developed the *LeapGesture Experiment* application that employs the Leap Motion controller [6]. The Leap Motion controller is a small motion sensing device (0.5 *cm* × 3.0 *cm* × 8.0 *cm*), composed of two video cameras and three infrared LEDs. It captures the human hand movements in a 60-80 *cm* hemispherical area, with a rate of approximately 200 frames per seconds. Data is streamed via USB to the Leap Motion software for image processing and information extraction. The Leap Motion API returns a set of frames; each frame contains lists of tracked hands, fingers, finger-like tools (pens, pencils, etc.), gestures, and movement data of the two hands [7]. Based on the observations from [2,3,7,18] we proposed a generic dataset consisting in both single-hand and bimanual free-hand gestures. Both types of gestures include geometric shapes, symbols, directional gestures and emblematic gestures. The purpose of this dataset is to have different types of free-hand gestures and to cover a wide range of tasks and experimental situations. The gesture set that we propose for one hand gesture commands is composed of:

- GEOMETRIC SHAPES, such as circle, square, triangle and ellipse; gestures are performed using the index finger;

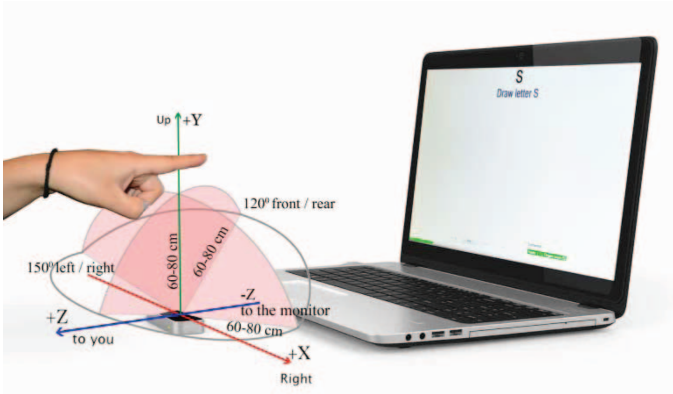

Fig. 6 The *LeapGestureExperiment* application (currently at version 1.0) using the Leap Motion controller: sensing area and system setup.

- SYMBOLS, such as question mark, C, S, E, W, and 2; these gestures are performed using the index finger;
- DIRECTIONAL GESTURES: up, down, left, right; the palm is open, facing upwards /downwards;
- EMBLEMATIC GESTURES, such as the victory sign and the thumbs-up gestures.

The bimanual gestures that we propose are:

- GEOMETRIC SHAPES, such as sphere, cube.
- DIRECTIONAL GESTURES: zoom-in, zoom-out, push-object, rotate clockwise; both hands make gestures to increase and decrease the size of an object, rotate that object, and push the object;
- EMBLEMATIC GESTURES, such as heart – make a heart using index and thumb fingers from both hands, pray –put the hands together for a short time, air quotes, applause, hand rubbing, this and that – move both hands simultaneously in right and left direction, listen up – both hand have palm open facing upwards and one hand start to hit the other hand open palm, etc.

When launching the *LeapGestureExperiment* application, the gesture dataset is loaded and the user sets the number of repetition for each gesture. Gestures are randomly presented to the user and stored in separate files using the XML format; for instance, the second execution for the gesture "circle" will be saved in the file "circle-2.xml".


```xml
<root>
 <Frame ID="429583" Timestamp="81448815214">
  <Hand ID="645" isValid="True" Handedness="Right" GrabStrength="0.00000" PinchStrength="0.00000" PalmWidth="87.51983">
   <Palm_position>( x = 52.59465 y = 167.56250 z = 191.09060 Pitch = 2.42170 Roll = 2.83745 Yaw = 2.87301) </Palm_position>
   <Palm_velocity>( x = 0.00000 y = 0.00000 z = 0.00000 Pitch = 2.35619 Roll = 2.35619 Yaw = 2.35619 )</Palm_velocity>
   <Palm_normal>( x = -0.43755 y = -0.89576 z = 0.07850 Pitch = -1.65821 Roll = -0.45438 Yaw = -1.74831 )</Palm_normal>
   <Direction>( x = -0.49992 y = 0.16977 z = -0.84927 Pitch = 0.19730 Roll = -1.89818 Yaw = -0.53203 )</Direction>
   <Basis>
    <xBasis>( x = 0.74742 y = -0.41084 z = -0.52209 Pitch = -0.66672 Roll = 1.06820 Yaw = 0.96106 )</xBasis>
    <yBasis>( x = 0.43755 y = 0.89576 z = -0.07850 Pitch = 1.48339 Roll = 2.68721 Yaw = 1.39328 )</yBasis>
    <zBasis>( x = 0.49992 y = -0.16977 z = 0.84927 Pitch = -2.94429 Roll = 1.24342 Yaw = 2.60957 )</zBasis>
   </Basis>
   <Arm>
    <Direction>( x = -0.45039 y = 0.21439 z = -0.86671 Pitch = 0.24249 Roll = -2.01507 Yaw = -0.47924 )</Direction>
    <WristPosition>(x = 85.85883 y = 159.45430 z = 231.04800 Pitch = 2.53752 Roll = 2.64766 Yaw = 2.78580)</WristPosition>
    <ElbowPosition>(x = 202.82520 y = 103.77620 z = 456.13540 Pitch = 2.91789 Roll = 2.04372 Yaw = 2.72319 )</ElbowPosition>
   </Arm>
   <Fingers Count="5">
    <Finger ID="6450" isValid="True" Type="TYPE_THUMB" Length="50.12876" Width="19.47771">
     <TipPosition>(x = -25.30100 y = 151.52710 z = 206.81890 Pitch = 2.50928 Roll = -2.97615 Yaw = -3.01986 )</TipPosition>
     <TipVelocity>(x = 0.00000 y = 0.00000 z = 0.00000 Pitch = 2.35619 Roll = 2.35619 Yaw = 2.35619 )</TipVelocity>
     <Direction>( x = -0.88648 y = -0.11344 z = -0.44865 Pitch = -0.24765 Roll = -1.44352 Yaw = -1.10227 )</Direction>
     <Bones>
      <Bone Type="TYPE_METACARPAL" Length="0.00000" Width="19.47771">
       <PrevJoint>( x = 62.46417 y = 160.85590 z = 248.97490 Pitch = 2.56798 Roll = 2.77119 Yaw = 2.89578 )</PrevJoint>
       <NextJoint>( x = 62.46417 y = 160.85590 z = 248.97490 Pitch = 2.56798 Roll = 2.77119 Yaw = 2.89578 )</NextJoint>
       <Direction>( x = 0.00000 y = 0.00000 z = 0.00000 Pitch = 3.14159 Roll = 3.14159 Yaw = 3.14159 )</Direction>
       <Center>( x = 62.46417 y = 160.85590 z = 248.97490 Pitch = 2.56798 Roll = 2.77119 Yaw = 2.89578 )</Center>
       <Basis>
        <xBasis>( x = 0.59457 y = 0.66900 z = -0.44602 Pitch = 0.98277 Roll = 2.41503 Yaw = 0.92720 )</xBasis>
        <yBasis>( x = -0.21074 y = 0.66498 z = 0.71651 Pitch = 2.39347 Roll = -2.83470 Yaw = -2.85554 )</yBasis>
        <zBasis>( x = 0.77594 y = -0.33202 z = 0.53636 Pitch = -2.58730 Roll = 1.16648 Yaw = 2.17562 )</zBasis>
       </Basis>
    ...
</root>
```


Fig 7. Data example excerpt from the output delivered by the *LeapGestureExperiment* application (currently at version 1.0) using the XML format.

A distinct identification is assigned to each participant. During the experiment, the user receives instructions about the gesture type and how to perform that gesture. Other information is available, such as the number of hands and the number of fingers detected by the Leap Motion controller, the total number of valid frames, and a progress bar that shows an approximate percentage of task completion. When the hands are positioned above the controller, recording starts. For each execution, we save the following data:

- FRAME – the current frame provided by the Leap Motion controller; we record the ID, timestamp and the number of hands;
- HAND – for each hand we store the ID, handedness, grab and pinch strength, palm width, palm position, palm direction, palm normal, and palm velocity;
- ARM - we collect the elbow direction and wrist position;
- FINGER –also linked to the hand: ID, type, length, width, tip position, tip velocity and tip direction;
- BONE – type, direction and center and 3D positions for finger joints.

Depending on the gesture type, one XML gesture file can have between 100 and 300 valid frames, with over 100 features per gesture. An example of an excerpt of an XML file is shown in Figure 7. This data can be used for training and testing the performance of gesture free-hand recognition algorithms or for constructing 3D hand visualizations.

VI. CONCLUSION

We presented in this paper three software tools that can help researchers, practitioners, and designers of gesture user interfaces to collect users' touch and free-hand gesture input. We tried to cover as many gesture input modalities as available today for mobile devices (*i.e.*, touch and free-hand input), for which we employed low-cost devices and sensors, such as the Leap Motion controller and the Ring Zero device. Our software tools (version 1.0 at the moment of this paper publication) are free to download and use by contacting the first author.

ACKNOWLEDGMENT

This work was supported from the project *"Gesture interfaces for visually-impairing interaction contexts"*, no. PN-II-RU-TE-2014-4-1187 financed by UEFISCDI, Romania.

REFERENCES

[1] D. Ashbrook and T. Starner, "MAGIC: A motion gesture design tool," SIGCHI Conference on Human Factors in Computing Systems, New York, NY, USA, 2010, pp. 2159–2168.

[2] C. Long Jr., "Quill: A gesture design tool for pen-based user interfaces," University of California, Berkeley, 2001.

[3] G. Cramariuc and S.G.Pentiuc, "Observing motor development of preschool children using depth cameras", The 11th Int. Scientific Conference eLearning and Software for Education Bucharest, April 23-24, 2015 10.12753/2066-026X-15-076, pp. 518-523

[4] Radu-Daniel Vatavu, Catalin Marian Chera, Wei-Tek Tsai. "Gesture profile for web services: an event-driven architecture to support gestural interfaces for smart environments," AmI'12, the Int. Joint Conf. on Ambient Intelligence. LNAI 7683, Springer, 2012, 161-176

[5] Ring Zero. http://ringzero.logbar.jp/

[6] Leap Motion. https://www.leapmotion.com/

[7] I.-A. Zaiti, S.-G. Pentiuc and R.-D. Vatavu, "On free-hand TV control: experimental results on user-elicited gestures with Leap Motion," Personal and Ubiq. Computing, 19, 5, pp. 821-838, 2015

[8] R. McCartney, J. Yuan and H.-P. Bischof, "Gesture recognition with the Leap Motion controller", Conference on Image Processing, Computer Vision, & Pattern Recognition, vol. 12, pp. 3-9, 2015

[9] Radu-Daniel Vatavu, Lisa Anthony, Jacob O. Wobbrock. "Gesture heatmaps: Understanding gesture performance with colorful visualizations," ICMI'14, the 16th ACM Int. Conf. on Multimodal Interaction. New York: ACM Press, pp. 172-179

[10] Radu-Daniel Vatavu, Lisa Anthony, Jacob O. Wobbrock. "Relative accuracy measures for stroke gestures," In Proc. of ICMI'13, the 15th ACM International Conference on Multimodal Interaction, 279-286

[11] Radu-Daniel Vatavu. "The impact of motion dimensionality and bit cardinality on the design of 3d gesture recognizers". International Journal of Human-Computer Studies, 71(4), 2013, pp. 387-409

[12] Lisa Anthony, Radu-Daniel Vatavu, Jacob O. Wobbrock. "Understanding the consistency of users' pen and finger stroke gesture articulation," GI'13, the 39th Graphics Interface Conference (Regina, Saskatchewan, Canada, May 2013). Toronto, Ontario: Canadian Information Processing Society, pp. 87-94

[13] Jacob O. Wobbrock, Meredith Ringel Morris, and Andrew D. Wilson. "User-defined gestures for surface computing," SIGCHI Conference on Human Factors in Computing Systems (CHI '09). ACM, New York, NY, USA, 2009, pp. 1083-1092.

[14] Meredith Ringel Morris, Jacob O. Wobbrock, and Andrew D. Wilson. "Understanding users' preferences for surface gestures," Graphics Interface (GI '10). Canadian Information Processing Society, Toronto, Ont., Canada, 2010, pp. 261-268.

[15] Jaime Ruiz, Yang Li, and Edward Lank. 2011. "User-defined motion gestures for mobile interaction," SIGCHI Conference on Human Factors in Computing Systems (CHI '11). ACM, New York, NY, USA, 2011, pp. 197-206.

[16] Radu-Daniel Vatavu. "User-defined gestures for free-hand TV control" Proc. of the 10th European conference on Interactive TV and video (EuroiTV '12). ACM, New York, NY, USA, 45-48, 2012.

Catalin M. Chera, Wei-Tek Tsai, Radu-Daniel Vatavu, "Gesture ontology for informing Service-oriented Architecture", IEEE Int. Symposium on Intelligent Control (ISIC), 2012, 1184-1189.

[17] Radu-Daniel Vatavu and Ionut-Alexandru Zaiti. "Leap gestures for TV: insights from an elicitation study", Proceedings of the 2014 ACM international conference on Interactive experiences for TV and online video (TVX '14). ACM, New York, NY, USA, 2014, 131-138.

[18] Yosra Rekik, Radu-Daniel Vatavu, Laurent Grisoni, "Understanding Users' Perceived Difficulty of Multi-Touch Gesture Articulation", Proceedings of ICMI'14, the 16th ACM International Conference on Multimodal Interaction (Istanbul, Turkey, November 2014). ACM Press, 2014, 232-239.

[19] Metrics & keylines – Layout – Google design guidelines, https://www.google.com/design/spec/layout/metrics-keylines.html

[20] [Ring SDK Android] How to join the public beta test, https://github.com/logbar/reference/wiki/%5BRing-SDK-Android%5D-How-to-join-the-public-beta-test

[21] Rekik, Y., Grisoni, L., Roussel, N. "Towards many gestures to one command: A user study for tabletops", Proceedings of INTERACT '13, Springer, 2013, 246–263. DOI=10.1007/978-3-642-40480-1_16

[22] Radu-Daniel Vatavu, Daniel Vogel, Géry Casiez, Laurent Grisoni, "Estimating the Perceived Difficulty of Pen Gestures", Proceedings of INTERACT'11, LNCS 6947, Springer, 2011, 89-106.