6DMG: A New 6D Motion Gesture Database

Mingyu Chen
mingyu@gatech.edu

Ghassan AlRegib
alregib@gatech.edu

Biing-Hwang Juang
juang@gatech.edu

School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332

ABSTRACT

Motion-based control is gaining popularity, and motion gestures form a complementary modality in human-computer interactions. To achieve more robust user-independent motion gesture recognition in a manner analogous to automatic speech recognition, we need a deeper understanding of the motions in gesture, which arouses the need for a 6D motion gesture database. In this work, we present a database that contains comprehensive motion data, including the position, orientation, acceleration, and angular speed, for a set of common motion gestures performed by different users. We hope this motion gesture database can be a useful platform for researchers and developers to build their recognition algorithms as well as a common test bench for performance comparisons.

Categories and Subject Descriptors

I.5 [**Pattern Recognition**]: General; I.4.8 [**Image Processing and Computer Vision**]: Scene Analysis—*motion, sensor fusion, tracking*

General Terms

Measurement

Keywords

6D Motion Tracking, Motion Gesture

1. INTRODUCTION

Human gestures can be broadly categorized as static, dynamic, or both, and they can be performed by hand, face, or body. Hand gestures are often the most frequently used and the most expressive. Among all human gestures, we are most interested in motion gestures, the dynamic gestures resulted from the user's body motion. More specifically, we focus on motion gestures of deliberate hand movements.

With the development of tracking technologies, motion-based control and motion gestures are gaining popularity and forming a complementary modality in human-computer interactions beyond the traditional devices (e.g., mouse, trackball, or touch-pad, etc.) and modes (e.g., touch and voice). The gaming experience of Wii, PlayStation 3 Move, and XBox 360 Kinect best exemplifies this idea. Motion gestures in game plays are usually limited in vocabulary size, and the recognition process normally follows certain scripts, to mimic the user's real-world motions and to trigger corresponding actions in the virtual world. When the player gets involved in the game, it is natural for him or her to follow the script that guides his or her motion, so as to comply with the motion recognition design in the game. Errors or ambiguity in motion recognition often can be resolved by limiting the possible options in the script.

When we extend the use of motion gestures from gaming to a general user interface, the system is expected to be able to handle a much larger set of motion gestures, which are likely to be rendered naturally with a substantial range of variability. As a result, the robustness and accuracy of the motion recognition algorithm become the primary challenge. Since high accuracy is desired for user satisfaction, it becomes very important to design a set of motion gestures that are large enough to support a general user interface and conform to natural human motions with customary semantic interpretations. These gestures can be displacement in position or orientation, and have speed, acceleration, or other kinematic properties. Different types of motion gestures may have discriminative features in different domains. Gesture recognition can be implemented using machine learning techniques, such as Hidden Markov Models [10, 15, 1], Finite State Machines [11], Dynamic Time Warping [8], data-driven template matching [20, 7], or feature-based statistical classifier [12, 6]. The reported correct recognition rates are above 90% on average. There is still room for improvement if expressions of motion gesture can be more rigorously formalized. The more precisely we can track the motion, the better recognition results we may achieve.

Although 2D tracking may be sufficient for general graphic user interfaces in the past, the time has come for a 6 DOF tracking capability. As human-machine interactions occur increasingly in virtual environments or through an emerging 3D user interface (3DUI), the interaction is more intuitive and immersive if we can control the position and orientation simultaneously, which requires 6 DOF motion tracking [18]. In such systems, we can capture 6D motion gestures with no extra cost since they already possess the required tracking capabilities, and the comprehensive tracking information comes as a side product which can be beneficial to the process of motion gesture recognition.

There is no standard or consensus on how motion gestures should be defined, performed, and mapped onto commands invoked on the system. Wobbrock [19] elicited motions from 20 participants in response to a variety of tasks to find the commonalities in mental models for user-defined gestures. The design of motion gestures can be considered as manipulating established motion taxonomies while preserving the logical (and conventional) mapping of causes to effects [13]. MAGIC [2] helps the designer create motion gestures with information on internal consistency, the distinguishability between gesture classes, and false positives. The design or logical function mapping of motion gestures is not in the scope of this paper; rather, we focus on the essence of the motion itself.

Similar to the case of speech recognition, it is desirable that the recognition system accommodates user-specific adaptation or customization, but it is also very important to achieve robust user-independent recognition. If we find a way to represent a motion gesture as a string of motion "alphabets", the recognition problem can be restated and solved in a manner analogous to automatic speech recognition. Therefore, we need a much better understanding of the motions in gestures, which arouses the need for a motion gesture database.

In this work, we present a database that contains comprehensive spatio-temporal data, including the position, orientation, acceleration, and angular speed, for a set of motion gestures performed by different users. Our contribution in this paper includes the rationale and analysis of a number of factors that may prove imperative in ensuring the sustainable usefulness of the database. The gesture set consists of widely used motion gestures such as circle, cross, v-shape, roll (wrist twisting), etc. It also has the most basic swiping motions which may be considered as the basic elements to form other complex gestures. Variations of the same gesture between individuals are expected, and recording motion gestures from different users ensures inclusion of the in-class variability. We would like to publish this database for people to derive, verify, and compare their recognition algorithms with it.

2. WHY 6D MOTION GESTURE?

The hand is a dexterous object with more than 20 degrees of freedom (DOF). Due to the interdependencies between fingers and joints, we can reduce the DOF, but studies have shown that it is not possible to use less than six dimensions [4]. In our case, the motion gestures are composed only by the location and orientation of the hand or the handheld device, i.e., 6D motion gestures.

Common motion gestures are mostly defined with 2D movements on a plane (usually the vertical plane), and we can perform recognition as if the motion gesture is captured with an image plane parallel to the motion plane. It is natural that human motions are still in 3D even though we intend to perform planar motions. Therefore, information other than 2D trajectory, such as depth and orientation, may give more insight into the motion gesture and improve the accuracy and robustness of recognition. For planar motion gestures, we can also derive the orientation of the motion plane, which can be useful in assigning directional meanings. Moreover, we are no longer limited to planar motion gestures if full spatial tracking results are available. Any type of motion can be considered as a gesture as long as it can be differentiated from others, and it is thus possible for designers and/or users to define their own motion gestures.

3. MOTION TRACKING

We have to capture the motion gestures before performing the recognition. Motion can be tracked via vision-based systems or via attaching sensors to the human's body. Based on images or videos, computer vision techniques for hand detection usually consider motion gestures as signals of 2D position and perhaps orientation. With the help of stereo or multi-view cameras, it is possible to track 3D position and orientation of the hand [17, 14]. Vision-based techniques provide more natural and unencumbered interaction. Xbox 360 Kinect demonstrates the tracking capabilities of human body in 3D, but it also has limitations on resolution and precision. For example, it cannot track subtle hand movements such as wrist twisting. Moreover, the accuracy and the robustness of vision-based systems are affected by many factors such as illumination and lighting, color of existing objects, and occlusion instances. In this work, our goal is to record accurate motions with as minimum noise introduced by the system as possible. Therefore, we use tracking devices to capture the precise position and orientation of the motion. Nevertheless, the generated motion database as illustrated in the remaining of the paper is still useful for vision-based recognition systems. One could consider our database as the ground truth hand motion.

There are several technologies for 3D motion tracking, such as optical-sensing, inertial-sensing, and magnetic-sensing [18]. These sensing technologies have their individual influence on the sampling rate, latency, spatial resolution, and spatio-temporal accuracy in implementation. Among them, we propose a hybrid framework that combines optical sensing and inertial sensing. The former measures the position of the optical tracker, and the latter estimates the orientation of the tracking device.

Optical motion tracking provides accurate estimation at a relatively high speed with small and untethered user-worn components. The tracking target can be either active / reflective markers or features extracted from video frames by vision-based techniques. A primary constraint of all optical systems is that there must be a clear line of sight between the tracking target and the optical sensor, and at least two pairs of the tracker-sensor relationships are needed for valid triangulation to determine the position of the trackers. The sampling rate of optical motion tracking depends on the frame rate of the cameras. 3D input devices usually have higher tracking noise compared to planar pointing devices, and are subject to hand tremor if held in space. More thorough study of the characteristics of spatio-temporal signals acquired by optical motion tracking has been done in [16, 3].

Comparing to optical motion tracking, inertial sensors have smaller latency and a much higher sampling rate. The accuracy of inertial sensing has been studied in static, quasi-static, and dynamic cases [5]. The inertial tracking device here refers to the Micro-Electro-Mechanical System (MEMS) accelerometers and gyroscope, which can be commonly found in the smartphone nowadays. The accelerometers measure the accelerations in the device-wise coordinates, and the gyroscope measures the angular speeds in yaw, pitch, and roll. The orientation, as a measurement, is actually accumulated from the angular speeds based on a global reference frame which can be determined by measuring the direction of gravity, and the magnetic north if magnetic sensors are equipped.

We can calculate the motion trajectory by integrating the accelerations along the corresponding orientation but cannot rely on its accuracy due to the drifting problem and error propagation over time. Hence, we use optical motion tracking for the trajectory and employ inertial tracking to supplement the orientation measurement. In addition to the 6 DOF of position and orientation, we actually have six extra dimensions from accelerations and angular speeds, which also infer the kinematic properties of the motion gestures.

In this work, we use WorldViz PPT-X4 as the optical tracking system, which tracks infrared dots at 60 Hz and transmits the tracking results with Virtual Reality Peripheral Network (VRPN) through Ethernet. It claims to have sub-millimeter precision and sub-centimeter accuracy with minimum 18 ms latency. As for the inertial sensors, we use the MEMS accelerometers and gyroscope embedded in Wii Remote Plus (Wiimote), which samples and reports at about 100 Hz. WiiYourself! library is used to communicate with Wiimote through Bluetooth.

4. MOTION GESTURE RECORDING

4.1 Recording System Setup

With the hybrid framework of motion tracking, we are able to record the motion gestures and build the database. We mount an infrared LED at the head of the Wiimote to fuse the two tracking devices together. The Object-Oriented Graphics Rendering Engine (OGRE) is used to render a controller with 6D motions captured by the hybrid framework, so as to provide the user with real-time feedback. We properly adjust the scale of the 3D model to make the rendered motion and the real-world action as close to one-to-one as possible. We further utilize the 3D display technology to provide better perception for the motion in depth.

The sampling rate is chosen to be 60 Hz, so the engine is set to update the motion data and render at 60 frames per second (fps). Even though the PPT system, Wiimote, and the render engine sample at different instants, we consider that they are synchronized with various delay and negligible latency jitter. The system continuously tracks the controller and updates the virtual controller on the display. However, only motions with explicitly signaled start and end points by the user are recorded. In our implementation, the user simply holds a button to start recording and releases it to stop. This button holding does not restrict the user from performing a natural and desired motion. Nevertheless, it is still possible that the recording is started or terminated early or late. Since this is also likely to happen in a real-world scenario, we consider the imprecise segmentation as part of the variation of the gesture data.

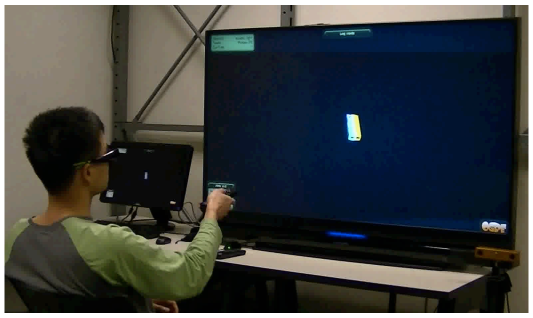


Figure 1: The experiment apparatus

During the gesture recording, we also shoot video asynchronized to the tracking devices. The camcorder is placed in front of the user and focuses on the hand motion. The supporting video is recorded at full HD resolution (1920×1080p). It serves the purpose for reference and can be used for vision-based analysis later. This contributes to having a complete database for the intended hand motions.

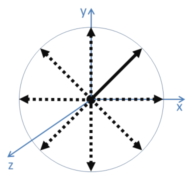
We also add a real-time playback function for the user to review the recorded trial of a motion gesture before it is stored into the database. We let the tester accept or reject the tentatively recorded motion gesture on the fly, which is more efficient than verifying the database offline at a later stage. After the user reviews and decides to save the recorded trial, the controller rumbles for 30 ms to confirm that the trial has been successfully inserted into the database. The trial number of the current gesture is also automatically updated based on the number of trials that have been saved.

The system always shows a virtual controller with one-to-one motion mapping to provide real-time visual feedback. A semi-transparent red overlay on the controller is shown while recording, and a semi-transparent green overlay indicates the system is displaying a playback. The current recording gesture name, tester, and the current trial number are also displayed. Figure 1 illustrates the gesture recording apparatus.

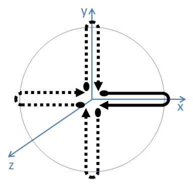
4.2 Motion Gesture Set

We consider swiping motions as the basic elements to form other complex gestures and hence include the group of swiping motions in eight directions into the gesture set as shown in Figure 2a. We also define a group of "poke" gestures that swipe rapidly forth and back in four directions (see Figure 2b). Other widely used motion gestures such as circle, cross, v-shape, wrist twisting (Figure 2c-2g) are included. Table 1 lists the full list of the names and statistics on durations of our 20 motion gestures. There are no "mirror" gestures, which means the direction and rotation are the same for both right- and left-handed users. New motion gestures may be added if needed in the future.

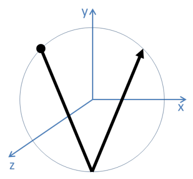
For every user, the gripping posture and the way he or she performs a given gesture can be different from one another. These variations are considered inherent in natural



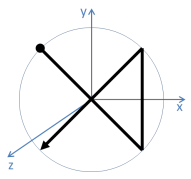
(a) Swipe



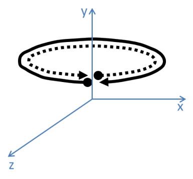
(b) Poke



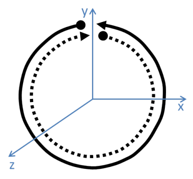
(c) Vshape



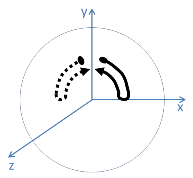
(d) Xshape



(e) CirHor



(f) CirVer



(g) Twist (Roll)

Figure 2: The illustration of gestures in our database

Table 1: The gesture list of 6DMG

Figure	Name	Duration (ms)	
		Avg.	Std.
2a	SwipeRight	866.2	345.2
	SwipeLeft	861.1	340.7
	SwipeUp	743.6	258.7
	SwipeDown	787.9	277.8
	SwipeUpright	754.5	282.7
	SwipeUpleft	748.9	291.4
	SwipeDnright	777.1	313.9
	SwipeDnleft	792.4	317.7
2b	PokeRight	1181.6	383.3
	PokeLeft	1242.4	418.4
	PokeUp	1206.4	389.9
	PokeDown	1183.6	415.6
2c	Vshape	1193.5	394.5
2d	Xshape	1655.2	466.1
2e	CirHorClk	1738.4	449.2
	CirHorCclk	1719.9	500.5
2f	CirVerClk	1806.5	549.3
	CirVerCclk	1707.7	532.1
2g	TwistClk	1054.8	315.5
	TwistCclk	1075.9	315.3

gesture rendering and need to be properly accounted for in the database, which is key to modeling and testing a user-independent motion gesture recognition task. We also would like to explore the differences between motion gestures performed by right-handed and left-handed users.

4.3 Recording Procedure

We recruited 21 right-handed and 7 left-handed participants (22 male and 6 female, and ranging in the age of 15 to 33) for recording. We first explain the basic functions of the motion controller to the tester, and let him or her play with the device for a while. Once the tester is familiar with the control interface, we start the recording process described as follows,

1. The tester resets the origin to a location he or she feels comfortable to start with.
2. The system selects a motion gesture from the set in the order as shown in Table 1, and briefly demonstrates it to the tester.
3. The tester presses and holds Button B during recording and releases the button upon termination of the trial.
4. After performing each trial, the tester reviews the playback and decides to save it or not.
5. After recording 10 trials, repeat Step 2 until all gestures in the set are recorded.

When recording, we advised the subject to perform the gesture in a consistent way, but we did not constrain his or her gripping posture, the gesture articulation style, range, and speed. The tester is asked to perform the soft calibration at the beginning of each trial. The hard calibration is required before starting a new set of gesture recording. The calibration processes will be explained later. The overall recording session per tester is less than one hour.

The accelerations and angular speeds of Wiimote are accessed through WiiYourself!, and we have to process the raw data as in Listing 1 for two reasons. First, the Wiimote coordinates need to be converted to the Ogre coordinates (see Figure 3). Second, we compensate the bias of angular speeds, where `bias` stores the estimated values of angular speeds when the gyroscope is static. After the conversion, we can compute the orientation. The Kalman filter has become the widespread orientation filter algorithms, but Kalman-based solutions demand a large computational load and high sampling rates, e.g., between 512 Hz and 30 kHz for human motion tracking. Hence, we implement the orientation estimation based on Madgwick's method [9], which is reported to perform similar and slightly better than the Kalman filter at relatively low rate with much lower computation.

We fuse the accelerations, i.e., the indication of gravity, to estimate the pitch and roll from the gyroscope. Unfortunately, Wiimote is not equipped with magnetometers, and we don't have yaw$_{mag}$ for automatic calibration in yaw. The orientation is updated in the background whenever a report is received from Wiimote, roughly every 10 ms, and our system fetches the orientation information at 60 Hz. Our

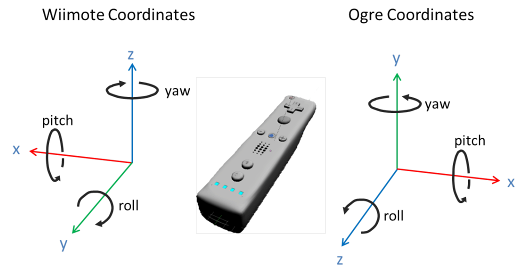


Figure 3: The coordinate conversion from Wiimote to Ogre

experimental results show that the orientation estimation is stable enough for the purpose of gesture recording. Also, all the processed accelerations, angular speeds, and `bias` in Listing 1 are available in our motion database if other techniques for orientation estimation are preferred.

Since the estimated orientation still suffers from the drifting issue and error propagation in yaw, we implement two methods for calibration. Hard calibration requires the controller to rest statically with the device coordinates aligned to the world coordinates for 5 seconds to estimate the bias (average) and the noise level (standard deviation) of the angular speeds in yaw, pitch and roll. However, the bias may still drift over time due to the heat issue, so we have to re-estimate it after a certain period of time. Soft calibration simply resets the orientation of the controller to identity quaternion (aligned to the world coordinates) and continues to accumulate the data measurement without bias estimation. Thus, the hard calibration is intended to tackle the drifting bias issue, and the soft calibration is the quick hack to eliminate the error propagation and drifting in yaw.

In addition to the orientation calibration, the user can reset the origin of the tracking position to any location that he or she feels comfortable to start with, even though the absolute position may not be a concern. We link the control functions above to the buttons on Wiimote for remote control.

5. 6D MOTION GESTURE DATABASE

In our implementation, a formal database structure is used to store the recorded motion gestures, which makes the management between gestures, testers, and trials very handy. The database structure can also be convenient for further development on motion gesture recognition. We use SQLite, a self-contained, serverless, zero-configuration, transactional SQL database engine for this purpose. The complete SQL database is contained in a single disk file, and the user can easily access it without installing any database server. All is needed is to include the SQLite library and a C++ header file that defines the structured type of samples and gestures. The SQLite schema of the database is shown in Listing 2. We also provide example programs: 6DMG loader and viewer to access and visualize the motion gestures (3D display is not required). To make the database portable and to keep the flexibility, we store the raw binary data. The data are also available in MAT format with the MATLAB exporter included in the 6DMG loader. The loader can be easily modified to export other specific file formats, such as AMC, BVH, C3D, or CSV.

Listing 1: Convert the raw Wiimote data (via WiiYourself!)

```
ogre.yaw   = -(wiimote.yaw  -bias.yaw);
ogre.pitch =  (wiimote.pitch-bias.pitch);
ogre.roll  = -(wiimote.roll -bias.roll);
ogre.acc_x = - wiimote.acc_x;
ogre.acc_y =   wiimote.acc_z;
ogre.acc_z =   wiimote.acc_y;
```

Listing 2: The schema of GestureTable

```
CREATE TABLE GestureTable(
    name   TEXT NOT NULL,
    tester TEXT NOT NULL,
    trial  INTEGER NOT NULL,
    length INTEGER NOT NULL,
    righthanded INTEGER NOT NULL,
    data  BLOB NOT NULL,
    noise BLOB NOT NULL,
    bias  BLOB NOT NULL,
    PRIMARY KEY (name, tester, trial)
);
```

Listing 3: The structure of samples and gestures

```
struct Sample{
    float timestamp;    // in ms
    float position[3];  // in meter
    float quaternion[4];
    float acceleration[3];  // in g
    float angular_speed[3]; // in rad/s
};
struct Gesture{
    string name;
    string tester;
    int trial;
    int length;
    int rightHanded; // 1:right, 0:left
    float noise[3];  // in degree/s
    float bias[3];   // in degree/s
    vector<Sample> data;
};
```

Listing 3 explains the data structure of the motion gesture in the database. All the 3-element arrays store data in the x-y-z order or in the yaw-pitch-roll order. Note that the orientation is stored in quaternion in the w-x-y-z order. Quaternions can be spherical linear interpolated without gimbal lock. Even though it is easier to interpret or visualize Euler angles, an Euler representation subjects to large and erratic errors if it reaches a singularity. Compared to rotation matrices, quaternions are more numerically stable and more storage-efficient.

The accelerations and angular speeds stored in each sample are the converted values as in Listing 1. In case a user wants to recover the raw angular speeds or derive the orientation measurement with different algorithms, we include the mean and standard variation of the angular speed of the hard calibration, i.e., bias and noise in Listing 2.

6. CONCLUSIONS & FUTURE WORK

In this work, we first explain the potential need and importance of 6D motion gestures, and then present a motion gesture database of comprehensive motion data, including the position, orientation, accelerations, and angular speeds, which is named "6DMG". The database itself is easy to manage and fully portable. 6DMG contains 20 motion gestures and 5600 gesture samples recorded by 28 participants.

With this motion gesture database, we plan to investigate motion gesture recognition using a hierarchical approach. It is no longer a symbolic classification problem, and we want to have a deeper understanding of motion gestures. As in speech recognition, we are interested in robust user-independent gesture recognition based on our 6D motion gesture database, which can improve the accuracy and the design space for motion gestures. Theoretically, both the displacement in position and orientation can be inferred from the accelerations and angular speeds. Our database provides both the explicit (position and orientation) and implicit (acceleration and angular speed) 6D motion data. The data can be useful to investigate motion gesture recognition with various dimensions of tracking signals. It is also an interesting signal processing problem to make direct use of raw data with the drifting issue.

We hope this motion gesture database can be a handy platform for researchers and developers to build their recognition algorithms and a common test bench for performance comparison. Moreover, a subset of information in our database, e.g., only the accelerations, can be used. The most recent release of the 6D motion gesture database and the accompanying example programs, including the viewer, loader, and exporter, are available at: `http://www.ece.gatech.edu/6DMG`

7. REFERENCES

[1] C. Amma, D. Gehrig, and T. Schultz. Airwriting recognition using wearable motion sensors. In *Proc. of the 1st Augmented Human Intl. Conf.*, AH '10, pages 10:1–10:8, 2010.

[2] D. Ashbrook and T. Starner. Magic: a motion gesture design tool. In *Proceedings of the 28th international conference on Human factors in computing systems*, CHI '10, pages 2159–2168, New York, NY, USA, 2010. ACM.

[3] M. Chen, G. AlRegib, and B.-H. Juang. Characteristics of spatio-temporal signals acquired by optical motion tracking. In *Signal Processing (ICSP), 2010 IEEE 10th International Conference on*, pages 1205 –1208, oct. 2010.

[4] A. Erol, G. Bebis, M. Nicolescu, R. D. Boyle, and X. Twombly. Vision-based hand pose estimation: A review. *Computer Vision and Image Understanding*, 108(1-2):52 – 73, 2007.

[5] A. Godwin, M. Agnew, and J. Stevenson. Accuracy of inertial motion sensors in static, quasistatic, and complex dynamic motion. *Journal of Biomechanical Engineering*, 131(11):114501, 2009.

[6] M. Hoffman, P. Varcholik, and J. LaViola. Breaking the status quo: Improving 3d gesture recognition with spatially convenient input devices. In *Virtual Reality Conference (VR10)*, pages 59 –66, Mar. 2010.

[7] S. Kratz and M. Rohs. Protractor3d: a closed-form solution to rotation-invariant 3d gestures. In *Proceedings of the 16th international conference on Intelligent user interfaces*, IUI '11, pages 371–374, New York, NY, USA, 2011. ACM.

[8] J. Liu, L. Zhong, J. Wickramasuriya, and V. Vasudevan. uwave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing*, 5(6):657 – 675, 2009. PerCom 2009.

[9] S. Madgwick. An efficient orientation filter for inertial and inertial/magnetic sensor arrays. Technical report, Department of Mechanical Engineering, University of Bristol, Apr. 2010.

[10] J. Mäntyjärvi, J. Kela, P. Korpipää, and S. Kallio. Enabling fast and effortless customisation in accelerometer based gesture interaction. In *Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*, MUM '04, pages 25–31, New York, NY, USA, 2004. ACM.

[11] S. Mitra and T. Acharya. Gesture recognition: A survey. *IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS - PART C*, 37(3):311–324, 2007.

[12] D. Rubine. Specifying gestures by example. *SIGGRAPH Comput. Graph.*, 25:329–337, Jul. 1991.

[13] J. Ruiz, Y. Li, and E. Lank. User-defined motion gestures for mobile interaction. In *Proceedings of the 29th international conference on Human factors in computing systems*, CHI '11. ACM, 2011.

[14] Y. Sato, M. Saito, and H. Koik. Real-time input of 3d pose and gestures of a user's hand and its applications for hci. In *Proceedings of the Virtual Reality 2001 Conference (VR'01)*, VR '01, pages 79–, Washington, DC, USA, 2001. IEEE Computer Society.

[15] T. Schlömer, B. Poppinga, N. Henze, and S. Boll. Gesture recognition with a wii controller. In *Proceedings of the 2nd international conference on Tangible and embedded interaction*, TEI '08, pages 11–14, New York, NY, USA, 2008. ACM.

[16] R. Teather, A. Pavlovych, W. Stuerzlinger, and I. MacKenzie. Effects of tracking technology, latency, and spatial jitter on object movement. *Proceedings of IEEE Symposium on 3D User Interfaces*, 9:43–50, 2009.

[17] A. Utsumi and J. Ohya. Multiple-hand-gesture tracking using multiple cameras. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 1, pages 2 vol. (xxiii+637+663), 1999.

[18] G. Welch and E. Foxlin. Motion tracking: no silver bullet, but a respectable arsenal. *Computer Graphics and Applications, IEEE*, 22(6):24 – 38, Nov 2002.

[19] J. O. Wobbrock, M. R. Morris, and A. D. Wilson. User-defined gestures for surface computing. In *Proceedings of the 27th international conference on Human factors in computing systems*, CHI '09, pages 1083–1092, New York, NY, USA, 2009. ACM.

[20] J. O. Wobbrock, A. D. Wilson, and Y. Li. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Proc. of UIST '07*, pages 159–168, 2007.