Dynamic Arm Orientation Detection During Body Motion as a Computer Interface

An Elegant Approach for a More Technological Age

Marcus Firmani

May 2, 2016

Abstract

Gestures serve an important role in human-computer interfaces, and advancing technology such as virtual and augmented reality makes 3-D gestures an increasingly important and powerful means of interaction. Robust gesture interaction depends upon reliable detection of arm position and hand gestures. Specifically, determining arm orientation, the position with respect to the user’s overall body position as it changes, is important because comprehension of the intentions associated with many natural gestures depends upon recognition occurring from the user’s frame of reference. Arm orientation is also important when differentiating different natural hand gestures using EMG signals. I will show how combining IMU data with simplifying assumptions based on an understanding of arm and body physiology and ergonomics can produce an efficient and accurate real-time dynamic arm orientation detection system that works during typical body movement.

Thesis Advisor: John Dougherty
Department of Computer Science
Haverford College
1 Introduction

In considering human-computer interfaces, one of the primary modes of interface has been one involving some kind of gesture, where a gesture consists of some combination of arm, hand, and finger motion. In the physical world, we primarily interact with things and accomplish tasks by physically moving things in the environment to achieve certain objectives. This makes a physical gesture-based interface with computers an appealing and potentially very intuitive means of accomplishing tasks with a computer.

Historically, many of the significant means of computer interaction have involved some form of gesture interpretation. A keyboard allows the user to perform simple finger motions which are translated into text input. A mouse enables the user to move, press, and manipulate items on a screen by translating intuitive hand and arm motions in a 2-dimensional plane into cursor motion on the 2-dimensional screen. Touchscreens remove some of the abstraction of the screen elements being manipulated by allowing the user to directly interact with the 2-dimensional view itself. Advances in touchscreen technology have even brought a limited third dimension of input capacity by discerning different levels of touch pressure, for example in the iPhone 6s screen, as well as in some iPads and other tablets. However, such interfaces are still fundamentally limited to operating on the environment represented on a 2-dimensional screen.

With the rise of wearable devices, virtual and augmented reality, and the internet of things, technology is expanding far beyond the scope of two-dimensional environments simulated on screens. As technology increasingly melds seamlessly into reality, it demands input methods that are less abstractions of how to navigate a virtual world and more intuitive means of interacting with the real one. Robust gesture
recognition in three dimensions will very likely be an important such interface. There are a number of devices on the market that are aimed at meeting this need, including a number of wearables such as the Myo armband used in this study. However, these devices often come with a relatively steep learning curve and unreliable translation of user intention into computer input. In order for an interface method to be useful, it must be intuitive and effective.

Reliably translating the wealth of data collected by wearable devices into accurate commands based on user motion is challenging, both due to the complexity of working in three dimensions and the fact that wearable devices such as the Myo are constantly receiving input. For the most part, more primitive interface technologies such as the mouse or touch screen consider most or all input they receive to be significant, because when the user is applying some form of touch or force, they are intending to interact with the device. By contrast, a wearable device is constantly receiving data and must be able to determine when the user intends to interact. Overcoming these challenges is critical to the success of 3-D interfaces and the technologies they facilitate.

1.1 Types of Interface Gestures

In considering the possibilities for gesture-based computer interfaces, it is important to get a comprehensive view of what qualifies as a gesture and how gestures may be translated into useful computer input. Karam and schraefel [sic] have generated “a taxonomy of gestures” that are used in human-computer interaction which can do just that. Within their taxonomy, there are four different categories: gesture style, application domain, enabling technology (input) and system responses (output) [6].

Within the gesture style category, deictic and manipulation gestures are especially relevant to a human-computer interface seeking to model itself on natural 3D interac-
Deictic gestures involve pointing to establish the identity or spatial location of an object within the context of the application domain," and "a manipulative gesture is one 'whose intended purpose is to control some entity by applying a tight relationship between the actual movements of the gesturing hand/arm with the entity being manipulated’" [6].

Deictic gestures may be “implicit in other forms of gestures, such as when pointing to identify an object to manipulate” [6]. A prime example is pointing with a computer mouse, which is typically associated with some form of manipulative gesture. However, in a 3-D context, the potential for genuinely independent deictic gestures may be much higher. Given the added dimension, the act of identifying a given point or region in the three dimensions available apart from any actual manipulation is potentially more critical.

Manipulative gestures may be performed in two or three degrees of freedom, in both cases providing a means of 2-D interaction and in the latter case potentially providing a means of 3-D interaction. In the case of 2 degrees of freedom, manipulations may be used “on the desktop in a 2-dimensional interaction using a direct manipulation device such as a mouse or stylus”, and for three degrees of freedom, “as a 3-dimensional interaction involving empty handed movements to mimic manipulations of physical objects as in virtual reality interfaces or by manipulating actual physical objects that map onto a virtual object in tangible interfaces.” [6] The latter two cases are of particular interest. These “empty handed movements” can be useful and intuitive to perform in some cases, but not always. The lack of tactile feedback and “boundary objects on which force may be exerted” [9] presents an issue in some cases. This issue affects some attempts at recognizing natural gestures made without the physical objects usually involved, because the gestures made in free space may be different than those made in a normal setting, or may be impossible to per-
form. Tanaka [9] gives musical performance as an example of such an interaction, where it can be difficult to perform some gestures used in playing a musical instrument without the physical presence of the instrument to provide the tactile response and resistance to enable the hand positions required. One potential solution in certain contexts would be to anticipate the gestures that are actually performed by an individual when attempting to simulate a gesture without the instrument or other substrate present. However, this is a limited solution because the number of possible gestures overall is being constrained, and so the risk of collisions increases. This means that it could become impossible to differentiate two or more fuzzy gestures from each other. Therefore, there is likely a fundamental limit to the interpretation of free-hand gestures that is more physiologically based, and this must be taken into account.

Both of these gesture styles in the real world depend to some degree on arm motion data as well as detailed finger motion data, and arm-based devices such as the Myo are well positioned to collect this data for processing and potential interpretation as computer input. The Myo already handles another gesture style Karam and Schraefel describe, namely semaphores, which are more symbolic or stylized gestures designed to communicate commands to a computer without exactly simulating any particular real-world action [7]. The built-in Myo gestures are essentially semaphores which may be associated with various commands. However, given the Myo hardware there is tremendous possibility open for the device to support deictic and manipulative gestures, as well.
1.2 Ergonomics of Gestures

While gestures have the potential to serve as a powerful and intuitive computer interface mechanism, such a mechanism can only work if the gestures interpreted are themselves intuitive and natural for the user. No matter how accurately a system can detect a given gesture, if it is uncomfortable or unintuitive to perform the gesture, the system will not be used. Conversely, because unnatural gestures are unlikely to be performed, it is less critical for a system to be able to discern them from other more natural gestures. This reduction of the space of gestures that need to be distinguished is important because it could make accurate recognition of useful gestures more feasible. Therefore, it is important to consider physiology both to ensure the recognized gestures are actually useful and to improve the recognition of those useful gestures. Two studies by Dounskaia et. al ([1], [2]) are of particular relevance for the physiology of the arms and resulting preferences in arm movement, and should be considered when designing interfaces that involve any form of motion of the arms.

When considering preferences in arm movement, we must determine whether the preferences are rooted in some fundamental constraint of the physiology of the body or are potentially avoidable in a virtual context. Specifically, in order to determine whether observed patterns in arm movement are actually related to physical constraints of the arms themselves, it is necessary to eliminate the possibility that these patterns are caused by our visual perceptions and conceptions of the world. Particularly for virtual reality interfaces, but also for standard computer interfaces, the context of the interactions could render some gestures more or less natural depending upon how different the context is from actual reality if movement preferences are based primarily upon our perceptions. The first study by Dounskaia et. al provides evidence that directional preferences in arm movements are not based on visual per-
ception of the spatial directions. The directional preferences were instead governed by the biomechanical forces involved in the actual arm movement performed. These preferences have significance when designing or anticipating arm gestures that are likely to be comfortable and natural for people to use for computer interaction. Because the movement preferences are not primarily based on perceptions of the world in which the user is interacting, the preferences cannot be ignored or altered by merely altering the perceptions provided to the user in the virtual environment. Rather, they likely represent a more fundamental constraint that must be accommodated when building a useful computer interface [1].

In order to accommodate physiological constraints on arm motion, it is necessary to understand exactly what those constraints are. The second study by Dounskaia et al. addresses this by investigating preferred arm movement patterns with respect to active versus passive motion in the various joints, particularly the shoulder and elbow. It supports prior findings suggesting that preferred arm movements tend to be ones in which one joint is active while the other is passive in moving the arm. This has consequences for the ergonomics of computer interface with the Myo especially, as predefined gestures that require non-preferred arm movements are less likely to be used or remembered, and could cause discomfort and strain for the user. Also, if a gesture is unlikely to be used, then it is not worth spending effort trying to discern whether the gesture is being performed as compared to a similar but more probable one [2].

1.3 Prior Detection Methods

After determining what arm motions will be natural for users to perform, we must be able to actually detect those motions. One method is to use arm-mounted accelerome-
ters and gyroscopes, as are present in the Myo. Such sensors can provide acceleration and orientation data in three axes, from which directionality of motion can be inferred. However, obtaining precise velocity and position data from the acceleration data requires single and double integration, respectively. This process typically produces a lot of compounded errors which render the data essentially useless for any period of time greater than a couple of seconds.

Gilbert et. al detail a successful approach for double-integrating acceleration data to determine position over a relatively long period of time (> 10 seconds) [4]. Usually errors accumulate so rapidly that such double integration does not offer useful position data. However, they found that a “critically damped double integrator” could significantly reduce these errors, leading to useful position data from accelerometer readings. Although this is interesting, the process may be excessively complex for a computer interface system using the Myo. In addition, it is unclear whether accurately determining precise arm or hand position in space at every moment is a useful feature for a gesture-based interface. Doing so would require the user to be constantly aware of their own arm position. With other interfaces such as the mouse or touchscreens, it is easy for the user to simply lift their hand off the device to stop interaction, allowing the user to rest or prevent unwanted inputs. It may be best to maintain some degree of separation between the interface and reality, particularly for standard desktop interaction. Also, from the users point of view, their arm’s absolute position is far less salient than its position relative to them in their own frame of reference. Therefore, for the purposes of an interface that is intuitive to use and effectively determines the user’s intent, it would actually be far more useful to be able to detect the arm’s position with respect to the user.

It is possible to use a machine vision-based approach that determines arm position using one or more cameras, as Han et. al demonstrated [5]. However, this method
depends upon the subject being within view of the cameras, which are typically in fixed positions, thereby limiting the mobility of the user. This method may also be excessively complex for many applications when the goal is to simply translate user intention based on arm motions in their reference frame.

Given that gestures often involve both hand and arm motion, we must consider how these motions may interact and the potential consequences of this interaction for detection and translation of gestures into viable computer input. One important method of detecting hand motions is via electromyographic (EMG) signals read from the muscles in the forearm. However, when the arm itself is moving, the forearm muscles also contract and produce EMG signals, potentially making it difficult to determine which signals are being produced due to hand gestures and which by arm movement, thereby preventing accurate translation of the signals into consistent computer input for a given gesture. Gazzoni et al outline prior work attempting to detect hand and finger movements and present the results of their study regarding discerning the movements of individual fingers based solely on EMG signals in the forearm. They address the complication of additional signal generated when the arm is moved/rotated during a hand movement [3].

Even when the arm is not actually moving, some of the muscles within it may be contracting depending upon whether the arm is being held in a certain position against gravity. Given that holding the arm in different positions could cause different muscles to contract, the EMG signals produced would likely differ. This adds a complication to the clouding effect when trying to detect and translate EMG signals produced by hand and finger movements. One study by You et al investigated just that, and they found that it was possible to predict individual finger motions using surface EMG data with a high level of accuracy, but that different arm positions resulted in different data. This indicates “that a robust finger motion decoding method cannot
be implemented without a scheme [that] detects changes of [the] arm’s posture.” This positional awareness could be achieved by a method like that presented by Gilbert et. al [4], or potentially by inferring positional changes from arm motion without maintaining a precise location of the arm in space. The Myo is very well suited to such data collection given that it can record both surface EMG signals and arm movements [10].

In conclusion, gestures are a powerful and ubiquitous means of human expression and action, and therefore are an important means of human-computer interaction. Emerging technologies demand that we make better use of gestures as a means of interfacing with them. In doing so, we must be mindful of the physical constraints of the body in designing interfaces that are comfortable and intuitive to use. Just because it is technologically possible to create an interface that can interpret certain gestures does not mean that such gestures will be useful or appealing for users. We must also be mindful of the technological constraints of gesture recognition. Just because a gesture is easy and intuitive for a human to perform does not mean that it will necessarily be easy to detect and interpret that gesture as computer input. Hand and finger motions in particular can be especially difficult to detect, particularly given interference caused by arm motions and position. A robust gesture-based interface must be able to detect and interpret intuitive and ergonomic motions consisting of as many combinations of arm, hand, and finger motions as possible. Therefore, it will be important to not only focus detection of arm movements on ones that are natural for the user to perform, but also to take the motions and associated positions into account when attempting to detect hand and finger motions.
2 Problem Definition

2.1 Arm Position Detection During Body Movement

Having established the importance of arm orientation and movement both as a form of gesture interaction and as a complication to hand gesture detection, we must consider the problem of detecting this arm orientation and motion. Detection in a static reference frame in which the user remains facing in one direction is relatively simple. In the absence of body motion, all motion detected can be attributed to the arm itself moving, and likewise positional changes also must be due to the arm being moved with respect to the body. However, in any situation in which the user may turn their body and move about, such assumptions will quickly render detection methods hopelessly inaccurate. Therefore, I will be focusing on detecting arm orientation (and to some extent, motion) with respect to the user during body movement. I am interested in arm movement both up and down and side to side, as well as rotational motion to some extent. The goal is to produce a means of determining user intentionality by focusing on isolating the arm’s position in the user’s frame of reference, and doing so in a computationally feasible way for real-time application with relatively limited computational power. Doing so will require making some simplifying assumptions about typical use conditions and incorporating observations about the physiology of human body motion. This is a similar approach to how automatic screen rotation works on most phones and other mobile devices. While there are situations in which it does not behave perfectly, such as when lying down, in most use case scenarios it is effective and intuitive. By assuming that users are usually sitting or standing upright, the proper orientation of the screen can be assumed to be with the bottom facing toward the ground in the direction of gravity, which can be easily determined with sensors in the device. In cases when this assumption is not valid, the behavior
of the device is still intuitive and easily understood by the user, who can then rotate the device in such a way that the screen will rotate as desired. This kind of intuitive melding with reality is what makes the system both efficient and effective, and will serve as the model for how the arm orientation detection system should function.

2.1.1 Arm Yaw Position

Perhaps the most important and most challenging aspect of arm orientation detection is determining the side-to-side position of the arm with respect to the user. From a human perspective, the world is primarily two-dimensional, with most objects differentiated by their direction in the horizontal plane in which we walk and live. At the same time, because we move about in this same plane, it can be difficult to isolate the movement and direction of the arm with respect to the body because the overall frame of reference frequently changes. Determining the user’s frame of reference with respect to the world is therefore an important objective. This study’s primary objective is to determine the yaw position in the user’s reference frame.

2.1.2 Arm Pitch Position

While horizontal orientation is perhaps most critical, there is still value in knowing vertical orientation, as well. Thankfully it is much easier to determine. Humans typically remain upright and on the surface of Earth, and so the frame of reference is fairly static. Also, the constant acceleration of gravity enables easy detection of the downward direction of the Earth with respect to the user.

2.1.3 Arm Roll Position

The roll of the arm is not directly related to the overall orientation of the arm, but is still an important value to consider with respect to the user’s frame of reference. For
example, the primary distinguishing feature between extending the arm to receive an
item and extending it to shake hands is the roll of the arm. It is also fairly simple
to determine the roll orientation with respect to the user, because the arm typically
rests in predictable roll orientations at frequent intervals, such as when the arm is
resting at one’s side.

3 Methods

3.1 Approaches

Determining arm orientation during body motion from sensor inputs is a tremen-
dously complex problem, primarily due to the challenges associated with determining
the yaw position. In addition, because we want to be able to determine orienta-
tion dynamically during motion using portable hardware, there are limitations on the
amount of processing that can reasonably be accomplished in real time. Therefore,
it is highly desirable to limit the complexity of the problem by making some reason-
able simplifying assumptions. These simplifying assumptions can be combined with
observations about arm physiology to develop various methods to assist in arm ori-
entation detection. I will first outline the various methods and the assumptions and
observations that inspired them. Then I will describe the way the methods interact
and work together as a coherent system.

3.1.1 Gravity-based Yaw Centering

Because the main reason that body rotation disrupts yaw-based arm position detec-
tion is that it alters the center yaw point of the arm, one simple means of correcting
the issue is to re-center the yaw on the new center point of the body after rotation.
Because the arm is roughly in the center of the yaw space when it is at a person’s side, the yaw can be accurately centered based on detection of the arm’s Z-position. Due to the constant acceleration provided by gravity on Earth, it is possible to reliably detect when the user’s arm is at their side by when the z-orientation in the world reference frame crosses a certain threshold. However, this method only works when the user lowers their arm to their side, and so does not help when the arm is raised for extended periods and body rotation occurs after the last centering occurred. It also is less likely to be useful when the user is sitting down if they do not fully lower their arm. However, there is less chance of the reference frame changing from body movement when seated anyway, so the other methods may be sufficient for such situations.

3.1.2 Yaw Boundary Crossing

Due to physiological constraints of both the arms and the human visual field, arm motions are primarily constrained to a region no wider than 90 degrees to either side of the center of the region the user is facing. Therefore, if the measured yaw exceeds this boundary on either side, it can be reasonably assumed that the body has likely rotated in the given direction, and thus the yaw can be recentered to account for this rotation. Because we cannot determine the exact combination of arm and body movement that produced the observed yaw shift, this method is not exact in its recentering ability. However, it does recenter the yaw on the “center” of the area that the user’s attention is being directed toward, and other methods can be used to fine-tune this initial recentering to more accurately reflect the physical center of the user’s frame of reference.
3.1.3 Velocity-based Directional Inference

When the arm’s position changes, the change itself can be detected via the direction of the velocity. However, rotation and movement of the body also causes the arm’s velocity to change. Ideally, the motion of the arm could be completely isolated from that of the body. However, a net velocity in one direction indicates the primary direction in which the combined forces of the arm and body are moving. For this reason, it may be useful to consider the direction of the velocity detected during motion to gain insight into the direction the user’s attention is being directed toward, even if it does not provide the absolute position of the arm. Additionally, if arm velocity is positive in a particular direction, the arm must be either moving in that direction independent of the body, or be stationary in the user’s reference frame and moving with the body. In either case, it will either stay fixed in the user’s frame of reference or move in the direction of velocity, but never in the opposite direction. Thus, velocity can be used to determine when a change in the arm position in the user’s reference frame has occurred and suggest where the arm is most likely to be in the user’s reference frame once motion has ceased.

3.1.4 Position Averaging

Over time, the average arm position value can provide some insight into the overall frame of reference and yaw center point. The usefulness of the average value likely decreases as body movement and rotation increases, but during periods of relatively little body motion it should tend to settle on some value. Due to the ergonomics of arm movement and typical rest poses, the value is likely between the center and directly across the body, similar to the case of pausing. Therefore, in combination with other methods to handle cases of body motion, averaging could enhance dynamic orientation
detection by fine-tuning the yaw center prediction after major body rotation that results in yaw boundary crossing.

3.1.5 Interaction of methods

Because different methods are optimized to work in different situations, the system must determine when to give each method precedence. When the arm passes a threshold for the z-position in the world reference frame (with respect to gravity), Gravity-based Yaw Centering takes precedence. Below this threshold, the absolute value of the velocity determines the precedence, with Velocity-based Directional Inference taking precedence when the velocity is over the threshold, and yaw center deviation taking precedence below it. If the previous direction of motion and average yaw guess agree on the direction, then the predicted orientation will move in that direction, otherwise the last saved yaw center will be used as the reference point. Yaw boundary crossing also triggers recentering whenever it occurs.

The interaction of the various methods can be modeled with the Finite State Machine (FSM) in Figure 1. The FSM is drawn in such a way that it can be visualized as being a ring of states surrounding the user. The states of the FSM correspond to the arm’s yaw orientation with respect to the user, and they are arranged and labeled to reflect the relative positions of the states with respect to each other and the user, who is represented by the diamond in the center. At any given time, there are four cardinal directions with respect to the user: front, right, back, and left. These are represented by the four states 1C, 2C, 3C, and 4C, respectively. Each C-state has corresponding L and R states representing left and right deviation from the given center state from the perspective of the user when facing that center state.

State 1C is the initial state, with its yaw value being fixed upon initialization. The yaw values for the other states are based upon this initial yaw value, with state
2C being 90 degrees greater, and so on. The yaw values are reset whenever the gravity-based centering method is activated. In such a case, the G transition will be taken to the appropriate C-state, and that C-state will have its center yaw value set to the current yaw. Then, the other states will base their center yaw values off of this new reference point. This can be visualized in the spatial FSM as a rotation of the machine around the user, with the states all remaining in the same places with respect to each other but the cardinal directions around the user being rotated with respect to the world reference frame.

The yaw values are also reset to the reference frame of the current yaw value whenever exceeding the outer yaw bounds for a given frame and therefore taking a LYB or RYB transition to an adjacent C-state. This will likely result in only a slight rotation of the machine in most cases given the frequency of updates occurring, as the user is unlikely to significantly exceed the yaw bound in a given clock cycle.

Transitioning from a C-state to its associated L-state or R-state depends upon yaw-based or velocity-based activation. Transitioning back to the C-state from its associated L- or R-state can occur from yaw-based or velocity-based activation or gravity-based centering. Exceeding the yaw bounds for the given frame causes a transition from the L-state to the C-state to the left, or from the R-state to the C-state to the right.
Figure 1: Finite State Machine Diagram
Table 1: FSM State Descriptions

<table>
<thead>
<tr>
<th>State Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#C</td>
<td>Arm in center of frame #</td>
</tr>
<tr>
<td>#L</td>
<td>Arm in left of frame #</td>
</tr>
<tr>
<td>#R</td>
<td>Arm in right of frame #</td>
</tr>
</tbody>
</table>

Table 2: FSM Transition Descriptions

<table>
<thead>
<tr>
<th>Transition Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LY</td>
<td>Left yaw (static or average)</td>
</tr>
<tr>
<td>RY</td>
<td>Right Yaw (static or average)</td>
</tr>
<tr>
<td>LV</td>
<td>Left Velocity</td>
</tr>
<tr>
<td>RV</td>
<td>Right Velocity</td>
</tr>
<tr>
<td>LYB</td>
<td>Left Yaw Bound Crossing</td>
</tr>
<tr>
<td>RYB</td>
<td>Right Yaw Bound Crossing</td>
</tr>
<tr>
<td>G</td>
<td>Gravity Centering</td>
</tr>
</tbody>
</table>

3.2 Implementation

3.2.1 Data Collection with the Myo

Data was collected using the Myo, an armband worn on the forearm which contains a 9-axis Inertial Measurement Unit (IMU) that provides orientation, angular velocity, and acceleration data, and EMG sensors which measure muscle activity, which can be correlated with specific hand gestures. Certain semaphoric hand gestures are natively recognized by the Myo software. [7] Data from the Myo was wirelessly transferred to a computer and collected every 10 milliseconds via Bluetooth.

3.2.2 Scripting in Lua

The data collected by the Myo was processed in real time on the receiving computer. The code was written in Lua, the scripting language used to write programs for the Myo. The code can be used to both report the raw data and to provide it as input to a game or other application. To reduce processing load, the data can be processed at
some regular interval which still provides sufficient information for typical interaction speeds used by humans.

### 3.2.3 Application to Game: Race the Sun

The code was tested in a modified version of a publicly available program enabling interaction with the game Race the Sun. [8] I modified the source code to make use of my dynamic arm position detection rather than its original static detection system based solely on yaw position. Because the game primarily makes use of left-right arm position detection, it presents a valuable use case for dynamic position detection. During gameplay, the user typically keeps their arm elevated for extended periods of time, so the gravity-based centering method will likely not be used often. The other methods must be able to effectively operate during these extended periods.

### 4 Experimental Results and Discussion

#### 4.1 Testing with the Race the Sun Game

Because the primary focus of this thesis was to address the issue of dynamic yaw orientation, it is important to determine whether the system I have developed actually improves side-to-side orientation recognition. I used a Myo script that enables user interaction with the Race the Sun game and retrofitted it to use my dynamic arm orientation detection in place of the original static yaw-based method. Due to the constraints of the game, I could not simultaneously feed it the original and dynamic data, so I simply ran numerous tests of both versions to compare observed accuracy of recognition, perceived responsiveness by those playing the game, and average scores achieved. I tested gameplay with users in both seated and standing positions, as well
as rotating in place to simulate real-world change in user reference frame orientation as would occur in a virtual or augmented reality system. To do this, I held the laptop computer that was running the game and moved it around the user as they rotated in place to follow the screen.

The tests were conducted as a blind experiment, with the subjects not knowing which code was being tested at any given time. The subject only knew which position they were in (seated, standing, or standing with rotation) and how the jet on screen responded to their arm motions. After each individual test, the subject was asked to rate the perceived responsiveness of the jet to the intentions they tried to convey through arm gestures, with the scale going from 1 to 10 where 10 was the most responsive.

After completing testing on the first two subjects, I decided to also investigate the effect of the velocity transitions within the dynamic detection approach, and so for the remaining three subjects I also conducted equivalent tests of the dynamic approach in which the velocity component was not considered.

As indicated in Figure 2, my full dynamic detection method with velocity transitions was perceived as more responsive than the static method by about 51.35% on average. The t-test probability was 0.00318, so the result is definitely significant. Somewhat surprisingly, it was also rated more responsive in all three body position cases, as shown in Figure 3. However, the t-tests indicate that the result is only significant in the rotating case (sitting t-test: 0.0887; standing t-test: 0.307; rotating t-test: 0.0159). Because the original goal was to improve responsiveness and eliminate yaw flipping during body rotation, and hopefully maintain the same level of accuracy and responsiveness during non-rotational use, this result is still very encouraging. However, these findings suggest that the dynamic approach may improve non-rotational responsiveness as well, making the approach even more appealing to
pursue. Additionally, all test subjects observed the yaw direction reversal during the rotation test and several had reversal issues even for the non-rotational tests when using the static detection method, while no reversal occurred when using the dynamic approach. Figure 4 shows the perceived responsiveness comparison for each subject, showing that all subjects preferred the dynamic method at least somewhat.

![Average Perceived Responsiveness](image)

Figure 2: Average Perceived Responsiveness

Comparing the responsiveness ratings for the three subjects who were tested on the dynamic approach without velocity transitions, removing the velocity transitions reduced the ratings compared to the dynamic approach with velocity included for both sitting and rotating, as shown in Figure 5. Two of the three individuals preferred the velocity version during the standing test, as well, but the third individual gave it a particularly low score which made it appear on average slightly worse. It is possible
that this was just an unusually bad run, so more testing would be needed before concluding that velocity should not be considered during standing.

These results suggest that even though the velocity transitions are not critical to the fundamental functioning of the system in that yaw reversal can be eliminated without them, they do contribute to the perceived responsiveness of the system to the user by helping take into account user intention more immediately than the yaw position may be able to. For rotation, including velocity seemed to improve the ability to discern body rotation and arm movement, as I had hoped. For instance, if the user rotates to the left and moves their arm to the right, their arm may not significantly alter its physical position in the current yaw frame. This means that purely yaw-based orientation detection might not be triggered. However, if there is sufficient net arm velocity to the right, then including velocity transitions would enable the user’s intention to be accurately detected despite the overall body rotation. Users reported
Figure 4: Perceived Responsiveness per Subject

that they didn’t need to move their arm as much for the system to recognize their intention when using velocity transitions.
As shown in Figure 6, the average scores for the two methods were essentially equivalent (t-test probability: 0.978303644), with the dynamic method very slightly outperforming the static method overall. This similarity may be due to the fact that the game involves some degree of luck, as well as the fact that the participants were not skilled in the game prior to the study. During the course of the study, the participants were quick to adapt to the feedback from the different methods. For instance, when the yaw direction would flip, they would often notice in time to adjust their arm motions to achieve their objectives in the game despite the change. However, the goal is not to test the adaptability of humans to inconsistent software, but rather to produce consistent and intuitive software that does not require such adaptations by the user. Thus, the perceived responsiveness and lack of yaw direction flips with the dynamic method is of primary interest.
Figure 6: Average Scores Achieved
5 Conclusion

The preliminary findings suggest that there is promise in the multi-faceted approach to dynamic arm orientation detection that I have developed. While I was developing it specifically with the Myo, the same principles could be applied to IMU data collected from any similar wearable device.

5.1 Future Work

5.1.1 Possible Extensions

It would be interesting to determine the average position of the arm empirically rather than assuming it is the center to improve the averaging approach. The velocity transitions were clearly important, but the specific threshold could be fine-tuned, as well. Also, the time increments at which the orientation is computed could potentially be reduced.

5.1.2 General Applications of Arm Position

As detailed when introducing this project, determining arm position has numerous potential applications as a means of computer input, particularly for virtual and augmented reality. I have produced a system that can be easily integrated with any existing or future projects that make use of arm orientation to provide more accurate data.

5.1.3 Enhancement of Hand Gesture Recognition

As with arm position itself, my system could be used to enhance hand gesture recognition by providing more accurate arm orientation information.
6 Bibliography

References


Appendices

A Source Code

The full source code can be downloaded at the following Github:

https://github.com/marcusfirmani/dynamyo

Due to the fact that Myo Connect does not have access to the file system, my code cannot be simply imported into another project. It must be copied into the files of the project intending to use the code.
B Retrofitted Race the Sun Code

The original Race the Sun source code is available here:

https://market.myo.com/app/5478fb74e4b05772f1d76920/race-the-sun-connector

In order to use the dynamic orientation detection system instead of the static one, include the mycode() and related methods referenced in Appendix A in the file and replace the current if/elseif/else case in the onPeriodic function that calls the fly methods with the following:

```java
local mycoderesult = mycode()
if (mycoderesult == "L") then
    flyLeft()
elseif (mycoderesult == "R") then
    flyRight()
else
    flyNeutral()
end
```