SAND2011-6328C

Reliable Forward Walking Parameters from Head-Track Data Alone

Category: Research

Abstract—Head motion during real walking is complex: The basic translational path is obscured by three gait-caused orthogonal bobbing components. Many VE applications would be improved if a bobbing-free path were available. This paper presents a method for identifying not only bobbing-free walk speed and bobbing-free walk direction, but also step frequency, step length, and step timings, using only the most commonly tracked body part: the head's position. We present two approximation methods for a forward-walking model: The Full-Model-Fitting Method is more complete, but more computationally expensive than the Expedited Method. We present results with each method. We validate these methods – comparing the method-calculated average step frequency, walk speed, and walk direction to averages derived from manually-tagged, head-track logs. We discuss how these methods could be used for animated self-avatars, Redirected Walking, and Walking-In-Place systems.

Index Terms—Virtual Environments, Locomotion, Head Tracking

1 Introduction

To accurately present view-dependent virtual environments (VEs), VE systems must track the user's head position. Due to cost and per-user time to don full-body trackers, many VE systems track only the head. Although many VE applications – including self-avatar animation, Redirected Walking, and Walking-In-Place (WIP) systems – could benefit from estimates of walking parameters such as walk speed and walk direction, walking-induced bobbing motions mask the true values (Figure 1). Other useful parameters such as step frequency and step timings are not directly available in systems that track only the head. In this paper, we present two methods that produce estimates for all of these values using only head-track position data.

The motion of the head while walking is governed by the biomechanics of bipedal locomotion [4, 12]. After basic translation, the most prominent characteristic of the head's motion during walking is *head bobbing* which has three signed, orthogonal components:

- Right-left bobbing (hereafter "rightward bobbing")
- Fore-aft bobbing (hereafter "forward bobbing")
- Up-down bobbing (hereafter "upward bobbing")

Although bobbing complicates generating a good estimate of the user's instantaneous speed and direction (Figure 1), it can aid in estimating step frequency and step timings.

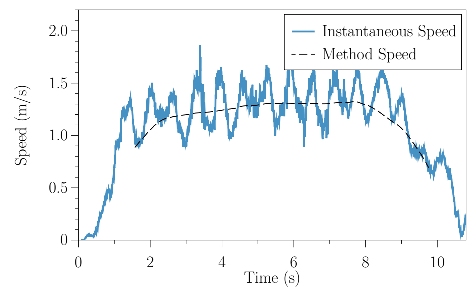
First, we describe VE applications that can benefit from reliable estimates of walking parameters. Then, we present the following:

- A mathematical model for forward-walking head movement,
- Two methods that approximate this model at real-time rates, and
- Validation of these methods against real-walking data.

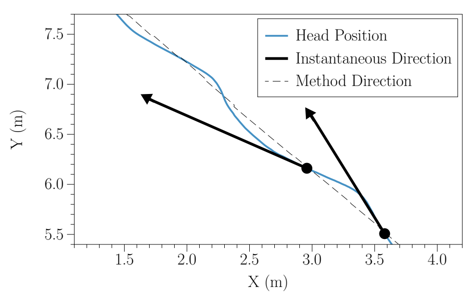
Animating Self Avatars: When comparing virtual locomotion interfaces, Usoh et al. [16] found that "presence significantly correlated with subjects' degree of association with their virtual bodies (avatars)." They suggested tracking the user's limbs so the avatar could match his motion. The introduction of the Ninetndo Wii, Microsoft Kinect, and similar gaming devices has added both user-body tracking and approximate self-avatar animations to gaming.

Real-walking-matching avatar motion has been demonstrated via full-body tracking systems, or by tracking a smaller number of key body parts. Chai and Hodgins [1] track seven positions on the user's body via simple video-based tracking, then employ a database including full-body-tracked motions to create – among others – plausible walking self-avatar motions. Liu et al. [9] employ local linear models to simplify a database containing full-body-tracked motions to six key marker positions. When those six positions are tracked, they animated full-body motions – including walking. Both systems require multiple trackers on locations throughout the body.

Plausible user-matching walking animations can be created with only head tracking by using our method's estimates of the user's position, walk direction, step length, and step timings.



(*a*) Walk Speed



(*b*) Walk Direction

Fig. 1. The instantaneous (*a*) horizontal walk speed (solid, lighter line) and (*b*) over-head-view of walk direction (thicker sold lines) compared to those obtained by our method (dashed lines). Head-bobbing-caused error in walk speed and direction hinders prediction of future location.

Redirected Walking: Redirected Walking and similar techniques enable users to walk through virtual worlds larger than the tracked space by disassociating the path traveled in the real world from the virtual-world path and imperceptibly redirecting users toward open space [11]. To properly redirect the user, the algorithm needs good estimates of current position, walk direction, and speed – parameters that our methods provide.

Interrante et al.'s [6] Seven League Boots is a related technique that requires approximating the user's bobbing-free walk direction to amplify the displacement in that direction. To estimate walk direction, they use a two-part approach using gaze direction as the bobbing-free direction when walking begins and a two-second-averaged direction as the user continues to walk.

Walking-In-Place: Walking-In-Place systems create virtual-world motions based on a user's in-place stepping motions [14]. Wendt et al.'s [17] GUD WIP system can create virtual walk speeds tuned to any user if a per-user walk-speed-to-step-frequency function is available. In their paper, they could create such functions only after manually tagging a user's head-tracked logs. Our method can create such functions automatically as a user walks.

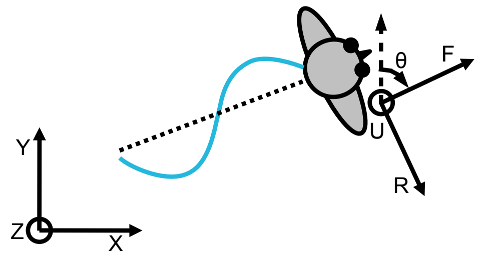


Fig. 2. The body-centric frame (r, f, u) is related to the world frame (x, y, z) by rotating around z by θ.

We also believe models similar to ours could be used to detect in-place steps and step frequency for WIP systems using only the head tracker.

Post hoc analysis: The methods presented herein could also improve post-hoc analysis of head-tracked locomotion logs resulting from comparative user studies (e.g. [18, 13]). For instance, when comparing the characteristics of different walking interfaces' paths, Whitton heavily smoothed Real Walking paths to eliminate head bobbing's effects [18]. This smoothing removes not only head bobbing, but also higher frequency components of the bobbing-free path. Our methods employ minimal smoothing to remove bobbing components – decreasing undesired smoothing.

2 The Forward Walking Model

The Forward Walking Model presented here is based on the biomechanics of human walking [4, 12]. A similar but simpler model was employed by Lecuyer et al. [7] to insert bobbing motions into keyboard or joystick-defined paths to simulate walking motions (see also [3, 15]).

Mathematically, forward-walking dynamic head positions ($\mathbf{h}(t)$) are the combination of the three bobbing components ($\mathbf{B_r}(t)$, $\mathbf{B_f}(t)$, $\mathbf{B_u}(t)$, for rightward, forward, and upward bobbing, respectively) and the remaining translation ($\mathbf{T}(t)$):

$$\mathbf{h}(t) = \mathbf{B_r}(t) + \mathbf{B_f}(t) + \mathbf{B_u}(t) + \mathbf{T}(t) \tag{1}$$

Each of the bobbing components can be estimated by a sinusoid:

$$\mathbf{B_v}(t) = \mathbf{v}(t) \cdot a_\mathbf{v}(t) \cdot \sin\left(2\pi f_\mathbf{v}(t) \cdot (t - t_0) + \phi_\mathbf{v}(t)\right) \tag{2}$$

where $\mathbf{v}$ is the bob's direction vector, $a_\mathbf{v}$ is its amplitude, $f_\mathbf{v}$ is its frequency, $\phi_\mathbf{v}$ is its phase, and t_0 is a centering term. Each varies with time.

The direction vectors for rightward ($\mathbf{r}$), forward ($\mathbf{f}$), and upward ($\mathbf{u}$) are three orthonormal vectors defining a body-centric walking space that is related to the world-centric forward direction in the horizontal plane by an angle (θ) (Figure 2):

$$\left[\ \mathbf{r}(t)\ \middle|\ \mathbf{f}(t)\ \middle|\ \mathbf{u}(t)\ \right] = \left[\begin{array}{c|c|c} \cos(\theta(t)) & -\sin(\theta(t)) & 0 \\ \sin(\theta(t)) & \cos(\theta(t)) & 0 \\ 0 & 0 & 1 \end{array}\right] \tag{3}$$

Equation 2 is constrained by biomechanics principles. The forward and upward bobs' frequency match the step frequency (f_s); the rightward bob moves at one-half the step frequency ($f_s/2$).

If bobbing is removed from Equation 1, only the forward translation ($\mathbf{T}$) remains. It is estimated as a bobbing-free head position ($\mathbf{c}$), and a forward speed (s):

$$\mathbf{T}(t) = \mathbf{c}(t) + \mathbf{f}(t) \cdot (s(t) \cdot (t - t_0)) \tag{4}$$

$$\mathbf{c}(t) = \left[\ c_\mathbf{x}(t) \quad c_\mathbf{y}(t) \quad c_\mathbf{z}(t)\ \right]^\mathrm{T} \tag{5}$$

Finally, in order to better fit curved walking paths, we include an angular velocity term ($\omega(t)$) with the walk direction, replacing $\theta(t)$ in Equation 3 with the following:

$$\theta(t) + \omega(t) \cdot dt \tag{6}$$

The full model of walking-induced head motion is found by expanding Equation 1 with Equations 2–6:

$$\begin{aligned} \mathbf{h}(t) = & \begin{bmatrix} \cos(\theta(t)+\omega(t)\cdot dt) \\ \sin(\theta(t)+\omega(t)\cdot dt) \\ 0 \end{bmatrix} \left(a_\mathbf{r}(t)\cdot\sin\left(\pi f_s(t)\cdot(t-t_0)+\phi_\mathbf{r}(t)\right)\right) + \\ & \begin{bmatrix} -\sin(\theta(t)+\omega(t)\cdot dt) \\ \cos(\theta(t)+\omega(t)\cdot dt) \\ 0 \end{bmatrix} \left(a_\mathbf{f}(t)\cdot\sin\left(2\pi f_s(t)\cdot(t-t_0)+\phi_\mathbf{f}(t)\right)\right) + \\ & \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \left(a_\mathbf{u}(t)\cdot\sin\left(2\pi f_s(t)\cdot(t-t_0)+\phi_\mathbf{u}(t)\right)\right) + \\ & \begin{bmatrix} c_\mathbf{x}(t) \\ c_\mathbf{y}(t) \\ c_\mathbf{z}(t) \end{bmatrix} + \begin{bmatrix} -\sin(\theta(t)+\omega(t)\cdot dt) \\ \cos(\theta(t)+\omega(t)\cdot dt) \\ 0 \end{bmatrix} \left(s(t)\cdot(t-t_0)\right) \end{aligned} \tag{7}$$

Different applications use various model parameters: Real-time walking avatars require step frequency (f_s), speed (s), direction (θ), step length ($l = s/f_s$), step timings ($\phi_\mathbf{r}$, foot-fall, and foot-off timings), and bobbing amplitudes ($a_\mathbf{r}$, $a_\mathbf{f}$, $a_\mathbf{u}$). Redirected Walking requires bobbing-free position ($\mathbf{c}$), speed (s), direction (θ), and angular velocity (ω). User-specific step-frequency-to-walk-speed functions for WIP require step frequency (f_s), and walk speed (s).

3 Approximation Methods

We provide two methods for approximating Equation 7 to estimate the Forward Walk Model's parameters from head-track data alone: A computationally-taxing Full-Model-Fitting Method, and an Expedited Method.

3.1 Full-Model-Fitting Method

This first method approximates all model parameters via the Levenberg-Marquardt Algorithm (LMA) – a numerical method for function minimization [8, 10]. Given a set of alterable parameters, LMA estimates the function's gradient with respect to those parameters. It iterates to find a local minimum employing a hybrid Gauss-Newton/Gradient-Descent Algorithm.

We employ the `lmfit` implementation of LMA [19] to fit head-track data ($\mathbf{h}(t)$) to our Forward Walking Model (Equation 7) by minimizing the error between a 3 second window of head-track data samples and the model-predicted head position, given the following LMA-modifiable parameters:

- Step frequency (f_s)
- Bobbing sinusoid amplitudes ($a_\mathbf{r}$, $a_\mathbf{f}$, $a_\mathbf{u}$)
- Bobbing sinusoid phases ($\phi_\mathbf{r}$, $\phi_\mathbf{f}$, $\phi_\mathbf{u}$)
- Walk direction (θ)
- Angular velocity (ω)
- Walk speed (s)
- Bobbing-free head position ($c_\mathbf{x}$, $c_\mathbf{y}$, $c_\mathbf{z}$)

3.1.1 Implementing the Full-Model-Fitting Method

Each 3 second data window has ˜150 head-track data samples – each including time (t_i) and position ($h_\mathbf{x}(t_i)$, $h_\mathbf{y}(t_i)$, $h_\mathbf{z}(t_i)$). Using the window's center time ($t_{0j} = (t_{n-1} - t_0)/2$), we seek a value for each LMA-modifiable parameter.

`lmfit` requires a method which accepts the LMA-modifiable parameters, and calculates the error ($\varepsilon(t_i)$) for each input datapoint. We use the two-norm (L_2) error between the model-predicted values (Equation 7) and the actual measurements as follows:

$$\varepsilon(t_i) = \sqrt{\varepsilon_{\mathbf{x}}(t_i)^2 + \varepsilon_{\mathbf{y}}(t_i)^2 + \varepsilon_{\mathbf{z}}(t_i)^2}, \text{ where} \tag{8}$$

$$\varepsilon_{\mathbf{x}}(t_i) = h_{\mathbf{x}}(t_i) - (c_{\mathbf{x}}(t_{0j}) + \cos(\tilde{\theta}(t_i)) \cdot \tilde{h}_{\mathbf{r}}(t_i) - \sin(\tilde{\theta}(t_i)) \cdot \tilde{h}_{\mathbf{f}}(t_i)) \tag{9}$$

$$\varepsilon_{\mathbf{y}}(t_i) = h_{\mathbf{y}}(t_i) - (c_{\mathbf{y}}(t_{0j}) + \sin(\tilde{\theta}(t_i)) \cdot \tilde{h}_{\mathbf{r}}(t_i) + \cos(\tilde{\theta}(t_i)) \cdot \tilde{h}_{\mathbf{f}}(t_i)) \tag{10}$$

$$\varepsilon_{\mathbf{z}}(t_i) = h_{\mathbf{z}}(t_i) - (c_{\mathbf{z}}(t_{0j}) + \tilde{h}_{\mathbf{u}}(t_i)), \text{ and} \tag{11}$$

$$\tilde{\theta}(t_i) = \theta(t_{0j}) + \omega(t_{0j}) \cdot (t_i - t_{0j}) \tag{12}$$

$$\tilde{h}_{\mathbf{r}}(t_i) = a_{\mathbf{r}}(t_{0j}) \cdot \sin\left(\pi f_s(t_{0j}) \cdot (t_i - t_{0j}) + \phi_{\mathbf{r}}(t_{0j})\right) \tag{13}$$

$$\tilde{h}_{\mathbf{f}}(t_i) = a_{\mathbf{f}}(t_{0j}) \cdot \sin\left(2\pi f_s(t_{0j}) \cdot (t_i - t_{0j}) + \phi_{\mathbf{f}}(t_{0j})\right) + s(t_{0j}) \cdot (t_i - t_{0j}) \tag{14}$$

$$\tilde{h}_{\mathbf{u}}(t_i) = a_{\mathbf{u}}(t_{0j}) \cdot \sin\left(2\pi f_s(t_{0j}) \cdot (t_i - t_{0j}) + \phi_{\mathbf{u}}(t_{0j})\right) \tag{15}$$

where $\tilde{\theta}$, $\tilde{h}_{\mathbf{r}}$, $\tilde{h}_{\mathbf{f}}$, and $\tilde{h}_{\mathbf{u}}$ are the model-driven approximations for direction, and the rightward, forward, and upward components of the head's position.

Although these equations suffice for a naïve fit, we found the following factors important in improving the final result's correctness: parameters' initial values, constraining parameters, steady-state calculations, and bobbing-fit improvements. We address each in turn.

Parameters' initial values: LMA finds a local minimum without guarantee that it be a global minimum. Therefore, the initial values provided to `lmfit` must be closer to the global minimum than to other local minima.

Through experimentation, we found that initial values for speed, direction, and step frequency most influence global minimum discovery. We initialize speed and direction with the mean values of the sampled speeds and directions on the 3 second data window:

$$s_0 = \frac{1}{n-1} \cdot \sum_{i=1}^{n-1} \sqrt{(h_{\mathbf{x}i} - h_{\mathbf{x}i-1})^2 + (h_{\mathbf{y}i} - h_{\mathbf{y}i-1})^2} / (t_i - t_{i-1}) \tag{16}$$

$$\theta_0 = \frac{1}{n-1} \cdot \sum_{i=1}^{n-1} \texttt{atan2}((h_{\mathbf{y}i} - h_{\mathbf{y}i-1}), (h_{\mathbf{x}i} - h_{\mathbf{x}i-1})) - \frac{\pi}{2} \tag{17}$$

where s_0 and θ_0 are the initial guesses for the speed and walk direction, and `atan2` is the C-language's method for the arc-tangent that preserves the angle's full range.

To estimate initial step frequency, the method counts vertical-bob crossings of the window-averaged vertical values. This coarse estimate is close enough for `lmfit` to yield the global minimum.

All remaining variables are initialized to the constraining range's middle value.

Constraining parameters: LMA minimizes in the parameter's gradient-vector direction and does not directly permit specifying permissible ranges for the parameters. However, the biomechanics of walking constrain several parameters' possible values. For instance, rhythmic walking step frequencies stay within 0.8 to 3.8 Hz. Table 1 lists the constrained parameters' permissible ranges.

To help `lmfit` keep variables within permissible ranges, we post-multiplied the error term (ε, see Equation 8) with a multiplier (m) – itself the sum of a multiplier (m_i) for each parameter (p_i):

Parameter	Units	Min	Max
$a_{\mathbf{r}}$	m	0.005	0.10
$a_{\mathbf{f}}$	m	0.005	0.08
$a_{\mathbf{u}}$	m	0.005	0.08
$c_{\mathbf{z}}$	m	see caption	see caption
f_s	Hz	0.8	3.8
s	m/s	0	6
ω	Radians/s	$-\pi$	π

Table 1. The minima and maxima allowed for various walking parameters. $c_{\mathbf{z}}$ is constrained to be within the middle two quartiles of the current data window's z-value range. All non-listed parameters have no constraints. These minimum and maximum possible values were chosen based on pilot data and from what may be found in the literature (summarized in [4]).

$$m = 1 + \sum_{i=0}^{\text{numParameters}} m_i \tag{18}$$

$$m_i = \begin{cases} 0 & \min \le p_i \le \max \\ (\min - p_i)/(\max - \min) & p_i < \min \\ (p_i - \max)/(\max - \min) & p_i > \max \end{cases} \tag{19}$$

This multiplier does not alter the parameters' gradients when they are within their acceptable ranges. For out-of-range parameters, the multiplier increases the gradient linearly with distance from the range – pushing them toward valid range.

Steady-state calculations: When the user walks at a steady pace, execution time can be improved by using inter-parameter relationships. We call this case "steady-state calculation".

We have discussed Full-Model-Fit functionality on any head-track data window. However, a single new piece of head-track data advances the window only one data point: Most of the window remains a good fit for the previously-estimated parameters. Furthermore, mathematical relationships permit maintaining some parameters' previous values.

Several model parameters are related by derivatives:

$$\frac{d\theta}{dt} = \omega \tag{20}$$

$$\begin{bmatrix} \frac{dc_{\mathbf{x}}}{dt} \\ \frac{dc_{\mathbf{y}}}{dt} \end{bmatrix} = \begin{bmatrix} -\sin(\theta + \omega \cdot dt) \\ \cos(\theta + \omega \cdot dt) \end{bmatrix} \cdot s \tag{21}$$

$$\frac{d\phi}{dt} = 2\pi f \tag{22}$$

The backward Euler approximations for Equations 20–22 relate previous parameter values (at $t_{0(j-1)}$) to current parameter values (at t_{0j}) as follows:

$$\theta(t_{0j}) = \theta(t_{0(j-1)}) + \omega(t_{0j}) \cdot dt \tag{23}$$

$$\phi_{\mathbf{r}}(t_{0j}) = \phi_{\mathbf{r}}(t_{0(j-1)}) + \pi f_s(t_{0j}) \cdot dt \tag{24}$$

$$\phi_{\mathbf{f}}(t_{0j}) = \phi_{\mathbf{f}}(t_{0(j-1)}) + 2\pi f_s(t_{0j}) \cdot dt \tag{25}$$

$$\phi_{\mathbf{u}}(t_{0j}) = \phi_{\mathbf{u}}(t_{0(j-1)}) + 2\pi f_s(t_{0j}) \cdot dt \tag{26}$$

$$c_{\mathbf{x}}(t_{0j}) = c_{\mathbf{x}}(t_{0(j-1)}) - \sin(\theta(t_{0j}) + \omega(t_{0j}) \cdot dt) \cdot s(t_{0j}) \cdot dt \tag{27}$$

$$c_{\mathbf{y}}(t_{0j}) = c_{\mathbf{y}}(t_{0(j-1)}) + \cos(\theta(t_{0j}) + \omega(t_{0j}) \cdot dt) \cdot s(t_{0j}) \cdot dt \tag{28}$$

where $dt = t_{0j} - t_{0(j-1)}$.

Using Equations 23–28 to replace $\omega(t_{0j})$, $\phi_{\mathbf{f}}(t_{0j})$, $\phi_{\mathbf{r}}(t_{0j})$, $\phi_{\mathbf{u}}(t_{0j})$, $c_{\mathbf{x}}(t_{0j})$, and $c_{\mathbf{x}}(t_{0j})$ in Equations 8–15 simplifies the error evaluation:

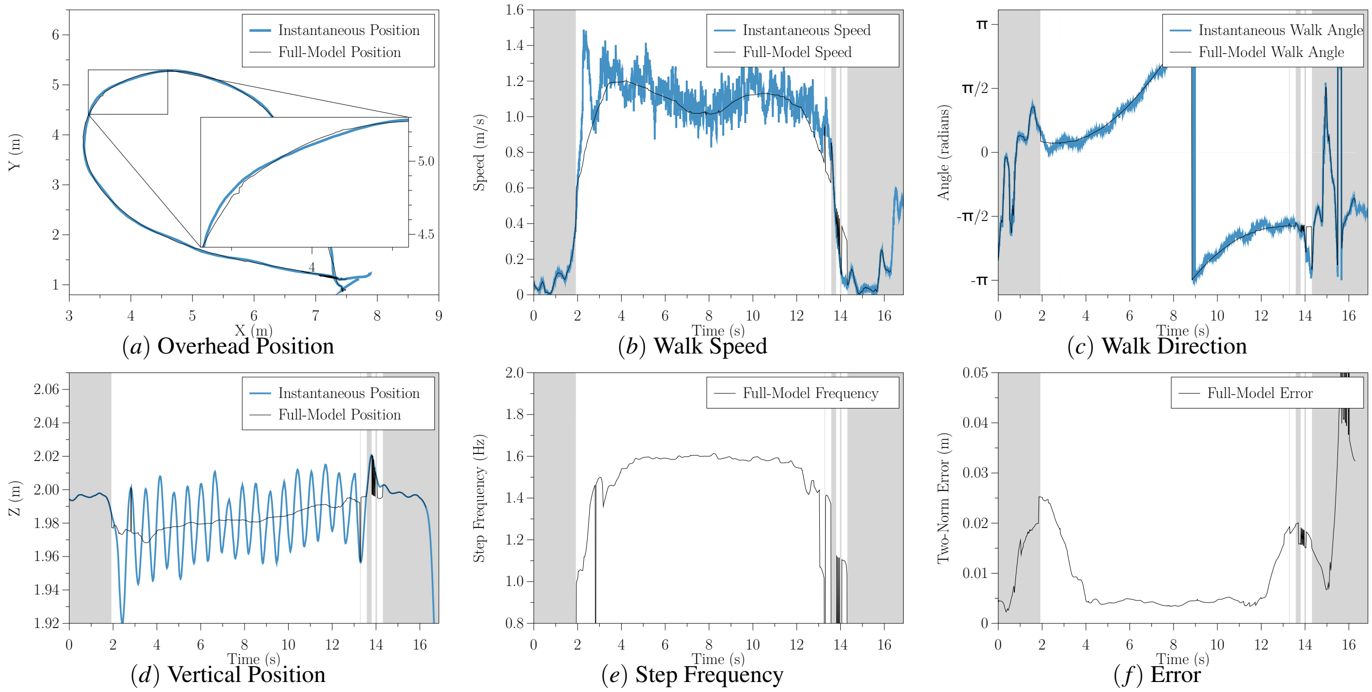


Fig. 3. Full-Model-Fit Results. User walked a loop. Thicker, lighter line: head-track data. Thinner, darker line: Full-Model-Fit Method. Background color: walking recognized when white. (a) Overhead view with magnification inset. (b) Horizontal walk speed. (c) Horizontal walk direction. (d) Vertical position. (e) Step frequency. (f) Two-norm error.

The walk direction, three phases, and two horizontal-position components are no longer free parameters – the previous iteration's results are used. After the simplified minimization completes, these six parameters are updated following Equations 23–28.

This steady-state enhancement speeds up `lmfit` execution by decreasing the number of free parameters, leading to fewer LMA iterations. However, the relationships stated in Equations 20–22 hold only so long as the angular velocity, step frequency, walk speed, and walk direction change only slightly. Larger changes indicate changes in walking pattern and require reverting to the non-steady-state, full-parameter solver. To avoid accumulating error, we reinitialize the steady-state four times per second by rerunning the non-steady-state solver.

Bobbing-fit improvements: Due to its relative simplicity, fitting the vertical component of the head's position ($h_{\mathbf{z}}$, Equation 11) was swift and robust. At times, we found that the horizontal components – two bobs, direction, and speed – went to local minima which did not match the real-walking values. To address this, we increased the weight of errors of the more robust vertical component, replacing Equation 8:

$$\varepsilon(t_i) = m \cdot \sqrt{\varepsilon_{\mathbf{x}}(t_i)^2 + \varepsilon_{\mathbf{y}}(t_i)^2 + \lambda \cdot \varepsilon_{\mathbf{z}}(t_i)^2} \qquad (29)$$

Setting λ to 80 leads to good convergence.

3.1.2 Full-Model-Fitting Method preliminary results

Implementation: We implemented this method as a single-threaded C/C++ system. When run on a Macintosh laptop with 2.13 GHz Intel Core 2 Duo CPU, 2 GB RAM, and Mac OS X 10.5.8, as people walk, it ran at roughly 50 data samples per second.

Figure 3 shows the results of running the Full-Model-Fitting Method against a sharply curved walking path. This case provides a demanding dataset which provides a good example of why the model needs to include angular velocity: As will be seen with the Expedited Method, a solely linear approximation does not fit as well. Figure 3(a): Angular velocity enables the method to stay within the original input data. Figure 3(b): The bobbing-induced speed changes are effectively ignored. Figure 3(c): The bobbing-induced direction change is removed. Figure 3(d): The bobbing-free vertical position cuts through the vertical bob. Figure 3(e): The step frequency is relatively stable at 1.6 Hz. Figure 3(f): The two-norm error is quite low at all times except during starting and stopping. The method works very well during rhythmic walking. Section 6 provides further validation of this method.

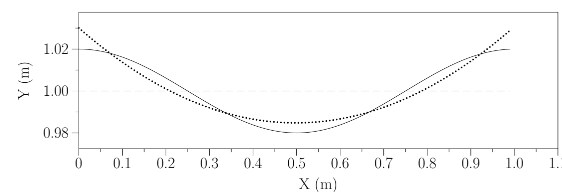


Fig. 4. Rightward-bob/angular-velocity error. This simulated, overhead view of a single rightward bob of walking data (solid line) demonstrates what errors arise when fitting the bobbing-free path to too little data. The dashed line shows the users' true bobbing-free path. However, with only one bob's data, an angular-velocity-including path (dotted line) mistakes the bob for the path.

3.1.3 Full-Model-Fitting Method discussion

This example illustrates several inherent model-fitting difficulties.

Non-orthogonal parameters: Many conceptually different walk-model parameters can provide similar results as each other:

- *Direction and rightward bob*: Figure 4 illustrates this point. When provided a data window containing only one full cycle of rightward bob, an angular-velocity-including fit leads to considerable walk direction errors. (By employing only angular velocity, we are fitting the horizontal head components to a circular arc.) Approximately 1.25 rightward-bob cycles appears to be a minimum to eliminate this effect. Because of this requirement and our lowest acceptable

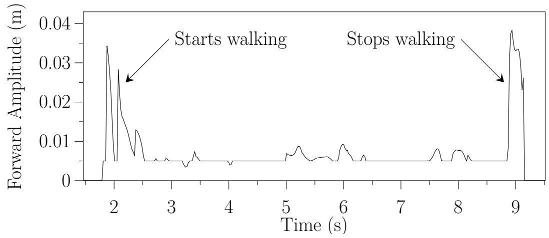


Fig. 5. The resulting forward bob amplitude for a user quickly walking a straight line at a constant pace. Note that LMA's resulting forward-bob amplitude increases dramatically during the brief starting and stopping periods.

step frequency (0.8 Hz), we use a 3 second window.

- *Acceleration and forward bob*: Changes in forward acceleration lead to forward-bob errors (Figure 5). LMA accounts for the acceleration-caused errors by increasing the forward-bob displacement.

Accumulating error: This error is best described by an example: During steady-state operation, the system updates the bobbing-free position by integrating a whole-window-fitting estimate of the angular velocity, walk direction, and walk speed (Equations 27 and 28). Even when these estimates are good fits for the whole window, they may not be good fits for the center time (t_{0j}). This problem occurs most often when step frequency, walk speed, and walk direction change considerably. Thus, when step frequency, walk speed, or walk direction change at a rate higher than our experimentally determined thresholds (respectively, 1.5 Hz/s, 1.2 m/s^2, 0.2 rads/s), we reset the LMA calculation to non-steady state.

3.2 Expedited Fitting Method

Some applications do not need all parameters provided by the model. Others find the Full-Model-Fitting Method's computational cost too high: To achieve 50 updates per second, it monopolizes one processor. Furthermore, since the model parameters are computed for the data window's central time (t_0), the Full-Model-Fitting Method's 3 second window leads to 1.5 seconds of latency. For many, this latency is too high. The Expedited Method trades fitting the complete model for computation at least 10 times faster than the Full-Model-Fitting Method, and approximately half the latency.

3.2.1 Implementing the Expedited Method

The Expedited Method is a two-stage process: First, a partial-model LMA fit is performed. Second, other parameters are approximated.

Stage 1 – LMA fit: The Full-Model-Fitting Method heavily weights the vertical head-bob component of the model (Equation 29). Its 3 second window size is required to avoid confounding angular velocity and the rightward bob model components (Figure 4). Therefore, by only fitting the vertical components of the model (Equations 11, 15, and 26), the Expedited Method can fit a 1.5 second window. Many of the Full-Model-Fitting Method's LMA-fitting enhancements still apply: Parameters must be properly initialized and constrained, and steady-state enhancements can be used.

Upon Stage 1 completion, we have good estimates for step frequency (f_s), step phase ($\phi_{\mathbf{u}}$), the bobbing-free vertical position ($c_{\mathbf{z}}$), and the vertical bob's amplitude ($a_{\mathbf{u}}$).

Stage 2 – Approximate parameters: Various of the remaining parameters can now be accurately approximated. We approximate walk speed (s), walk direction (θ), and the horizontal position ($c_{\mathbf{x}}$ and $c_{\mathbf{y}}$).

The principal confounder in estimating each of these parameters is head bobbing. For such sinusoids, the following equation is true for any frequency, amplitude, or phase, and for any integer n:

$$0 = \int_{i}^{i+n\cdot\frac{1}{f}} a\cdot\sin(2\pi f\cdot x+\phi)dx \tag{30}$$

The integral of any sinusoid over an integral number of periods is zero.

Therefore, if $\bar{X}_{t_{\text{init}}}^{t_{\text{fin}}}$ is the arithmetic mean from time t_{init} to t_{fin}, we estimate each of these bobbing-free parameters as follows:

$$\theta = \bar{X}_{(t_n-\frac{2}{f_s})}^{t_n}\left(\texttt{atan2}((h_{\mathbf{y}i}-h_{\mathbf{y}i-1}),(h_{\mathbf{x}i}-h_{\mathbf{x}i-1}))\right)-\frac{\pi}{2} \tag{31}$$

$$s = \bar{X}_{(t_n-\frac{2}{f_s})}^{t_n}\left(-\sin(\theta)\cdot(h_{\mathbf{x}i}-h_{\mathbf{x}i-1})+\cos(\theta)\cdot(h_{\mathbf{y}i}-h_{\mathbf{y}i-1})\right) \tag{32}$$

$$c_{\mathbf{x}} = \bar{X}_{(t_n-\frac{2}{f_s})}^{t_n}\left(h_{\mathbf{x}}\right) \tag{33}$$

$$c_{\mathbf{y}} = \bar{X}_{(t_n-\frac{2}{f_s})}^{t_n}\left(h_{\mathbf{y}}\right) \tag{34}$$

where t_n is the time of the most recently received data sample. The mean over two step periods is used because the rightward bob occurs at one-half the step frequency.

Two time samples: Stages 1 and 2 use different size data windows to estimate their results: Stage 1 uses a fixed-width 1.5 second window; Stage 2 uses a variable-width 2 step-period window. The parameters are estimated at two different times. Thus there are two latencies: one is fixed (0.75s); the other varies (one step-duration). At step frequencies above 1.33Hz (a modest walking pace), the Stage 2 parameters' latency is lower than the Stage 1 parameters.

Remaining parameters: The model parameters not estimated by the Expedited Method are the angular velocity, and the horizontal bobbing sinusoids' amplitudes and phases.

3.2.2 Expedited Method results

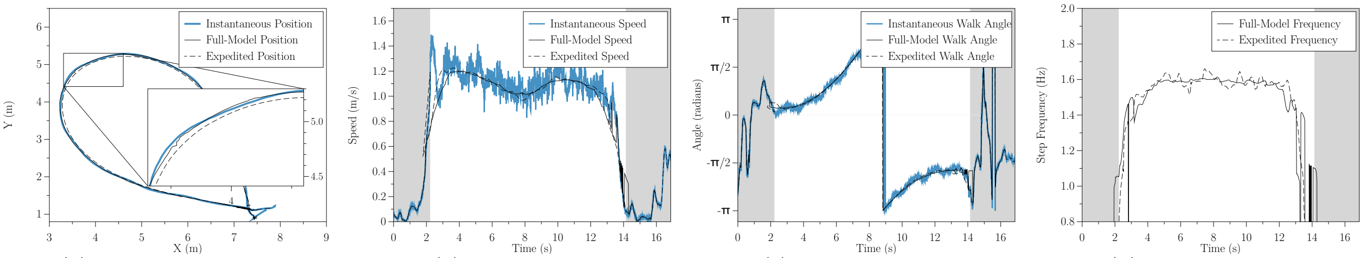
Implementation: When run on the same system configuration, and against the same input data as the Full-Model-Fitting Method, the Expedited Method ran ˜10 times faster.

The Expedited Model's estimated parameters' results closely approximate those of the Full-Model-Fitting Method. Figure 6 shows some of these. Figure 6(a): During hard turns, the linear approximation of the path is unable to stay within the curved path's bobs. Figure 6(b) and (c): Speed and direction are very similar to the Full-Model-Fitting Method's results, and the smaller window size better matches the acceleration and deceleration periods. Figure 6(d): Because of its smaller window size, the Expedited Method's frequency varies more.

3.2.3 Expedited Method discussion

Since the second stage's window size depends on the first-stage-estimated step frequency, two issues arise:

- The second stage parameters' center time does not monotonically increase. When the frequency decreases, the second-stage window size increases – moving the window's center time. If this window-size increase is large enough, the second-stage time can move backwards. This effect is visible in Figure 6(b) at approximately 2 seconds.
- When the first-stage model fit converges to an incorrect minimum, all second-stage parameters are inaccurate. This occurs most frequently when the user is not walking. Therefore, we added a Walking-Recognition Method, trusting second-stage results only when walking is recognized.



(a) Overhead Position (b) Walk Speed (c) Walk Direction (d) Step Frequency

Fig. 6. Expedited Method results compared to Full-Model-Fit Method. User walked a loop. Thicker, lighter line: head-track data. Thinner, darker line: Full-Model-Fit Method. Dashed-line: Expedited Method. Background color: walking recognized when white. (a) Overhead view with magnification inset. (b) Walk speed. (c) Walk Direction. (d) Step Frequency. Note that although the Expedited bobbing-free position is somewhat less accurate than the Full-Model's, the bobbing-free directions are nearly identical.

3.3 Recognizing Walking

When the user is walking, the vertical head bob matches a sinusoid very well. When the user is not walking, his vertical head position is largely unchanging. These observations provide the following Recognizing Walking heuristic.

To recognize periods of walking, we perform two LMA fits to the vertical data: a linear fit, and a sinusoidal fit. Both the Full-Model-Fitting and Expedited Methods already perform the sinusoidal fit, and the linear fit (with only two parameters) is fast.

When the error for the vertical sinusoid fit (Equation 11) is lower than the linear fit's error *and* all vertical parameters are within real-walking ranges (Table 1), the system recognizes the user as walking. When the user is not walking, the window-centered value of all instantaneously-approximable parameters ($\mathbf{c}$, s, and θ) are used. When the user is walking, all method-approximated parameter values are output.

The results of this heuristic are shown with a white background in Figures 3, and 6. This method recognizes walking within the first half step and stops within the last half step. Some false negative blips are visible as spikes in the output in Figure 3. When walking is not recognized, both methods revert to the instantaneous value for position, speed, and direction; and to zero for frequency, and step timing. The jagged-looking outputs seen in the figures could be filtered by any application requiring real-walking data.

4 Comparison With Existing Systems

We know of no existing systems that provide high-frequency (50 Hz) updates for walking parameters. In Exercise Science, commercial systems exist that provide estimates of some of these parameters at 1-3 Hz (e.g. [5, 2]). For instance, to estimate step length and step frequency, these systems employ trackers near the feet, and pressure sensors under each foot to identify foot-fall and foot-off location and events. Estimates are available only after two consective foot-fall events and update only with later foot falls.

Interrante et al. [6] estimate a bobbing-free walk direction through fixed-width averaging over a 2 second window – similar to our Equation 31. This leads to higher direction errors and latency than our methods. Their fixed window size – not an integral number of steps – generally includes a non-zero amount of all bobs (see Equation 30). The effect of this error declines as step frequency increases. With a 2 second window, the latency in their estimate is greater than that of the Expedited Method for all step frequencies above 1 Hz.

5 Fourier Techniques

We attempted a Fourier-based approximation method before adopting LMA. However, the Fourier-based methods were considerably less accurate at matching step frequency than the LMA-based methods. The following is a brief description of our findings. **NOTE: We could provide a citation to further details, but do not to preserve double-blind.**

If one walks at a fixed step frequency (f), his head's vertical trace is approximately a sinusoid. Transformed to frequency domain, the trace becomes two spikes ($+f$ and $-f$). Windowing the signal in the spatial domain multiplies it by a box function – equivalent to convolving the frequency domain representation with a sinc function, with the sinc function's maximum height at the original spike's location.

Narrower box filters (with lower latency and less smoothing) lead to wider sinc functions – causing the $-f$ sinc function's magnitude to be non-negligible at the $+f$ sinc function's center. Given perfect sine inputs, at 1.2 cycles per window, errors are on the order of 0.2Hz. Even at 3.0 cycles per window (a four-second window size for lower step frequencies), errors are on the order of 0.05Hz.

Given the same data, Levenberg-Marquardt minimization produces errors below 10^{-6}Hz.

6 Validation Against Real-Walking Data

Lacking a previous system that provides the same parameters at similar frequencies as our methods, we had to validate the output of our methods against averaged results from manually tagged windows. As part of their user study, Wendt et al. gathered and manually tagged five-step regions in head-track logs of users really walking [17]. We validated the approximation methods' outputs against average step frequency, walk direction, and walk speed from this dataset.

6.1 Recap of Method of Wendt's User Study

Design: Each subject walked an approximately 9m straight-line path four times at each of four prescribed stepping frequencies. The frequencies were ordered using Latin Squares; each was executed before any repeated.

Participants: There were eight paid participants – all male, all with unimpaired walking, ages 19-29.

Procedure: Subjects wore a 3rdTech HiBall-3100^{TM} head tracker. Subjects practiced walking in time to a metronome (1.67 Hz) then began their trials. The metronome's speed during trials was 1.0, 1.5, 2.0, or 2.5 Hz.

Measures: The head's vertical position was graphed and manually tagged to identify periods of consistent walking – periods containing five complete vertical head-bobbing cycles (Figure 7). We estimated three "ground truth" average values over these five-step periods: average speed and direction were calculated following Equations 31 and 32; average step frequency was estimated as five steps divided by period duration.

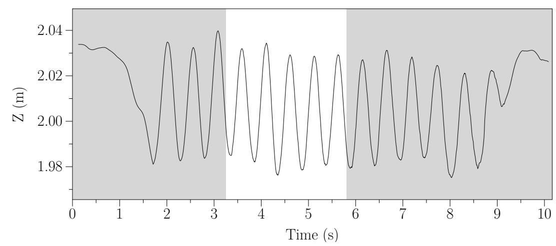


Fig. 7. Vertical component of head-track data for one trial. The white background indicates the manually-tagged five-step period.

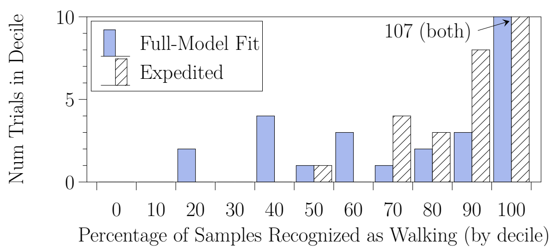


Fig. 8. Each trial recognized some percentage of the samples on the manually-tagged window (Figure 7) as walking. Each method performed perfectly – recognizing all samples as walking – on 107 of the 123 trials (87%). To illustrate, the 20% decile shows that two Full-Model-Fit Method trials recognized between 20-29% of the windows' samples as walking. The Expedited Method trended toward better scores on the 13% of the trials that imperfectly identified walking.

For this validation, we also ran the Full-Model-Fitting and Expedited Methods against these data logs to estimate Forward Walking Model parameters.

6.2 Analysis and Results

Five trials were lost from the original dataset due to system errors. No subject lost more than one trial.

Accuracy of Recognition of Rhythmic Walking: Before valid estimates can be obtained for step frequency, walk speed, and walk direction, the methods must converge to a walking-recognized state (Section 3.3). The number of method-recognized-walking samples was divided by the number of samples in the manually-identified window – yielding a per-trial percentage of correctly identified samples. A perfect approximation method would recognize 100% of the samples as walking. Figure 8 splits the trials by their percent-walk recognized deciles. Each of the estimation methods recognized walking perfectly in 107 of 123 total trials (87%). For the 13% of the trials that imperfectly recognized walking, the Expedited Method trended toward higher percentage-correct recognition scores.

Estimated Parameters: We averaged the methods' estimates for step frequency, walk speed, and walk direction over the manually-tagged windows. We compared these averaged, method-generated values (val_{mg}) to the "ground truth" averaged values (val_{gt}). For walk direction, we used the absolute error (in degrees). For step frequency and walk speed, we used the percent error ($\text{err}_{\%}$):

$$\text{err}_{\%} = \frac{\text{abs}(\text{val}_{\text{gt}} - \text{val}_{\text{mg}})}{\text{val}_{\text{gt}}} \tag{35}$$

Table 2 reports the results. We divided the trials into two groups: perfect walking-recognized scores (100%), and imperfect scores (<100%). We report the error scores of the fiftieth, ninety-fifth, and hundredth percentiles. These three percentiles demonstrate the errors' trends. When they recognized walking perfectly, each method gave

Method	Perc. Walk. Recog.	n	Metric	Units	50%	95%	100%
Full-Model	100%	107	Freq.	%Err	0.6	2.0	3.1
			Speed	%Err	1.5	4.4	8.2
			Dir.	deg	0.6	1.7	5.2
Expedited	100%	107	Freq.	%Err	0.5	1.7	3.1
			Speed	%Err	1.2	4.0	6.2
			Dir.	deg	0.5	1.1	2.3
Full-Model	<100%	16	Freq.	%Err	6.3	13.7	31.5
			Speed	%Err	2.6	7.0	83.9
			Dir.	deg	0.6	2.3	3.4
Expedited	<100%	16	Freq.	%Err	3.3	9.5	18.8
			Speed	%Err	1.8	4.1	5.3
			Dir.	deg	0.6	1.7	5.72

Table 2. Methods' step frequency, walk speed, and walk direction errors compared with "ground truth" averages at the 50^{th}, 95^{th}, and 100^{th} percentiles. Results are split between perfect walking-recognized trials (100%), and imperfect (<100%). To illustrate, the top row shows that of the 107 perfect walking-recognized trials for the Full-Model-Fitting Method, 53 had error less than 0.6%, 99 had error less than 2%, and the worst had 3.1% error.

highly accurate values almost uniformly. Even when failing to perfectly recognize walking, the reported values were quite good (less than 10% error) in about 95% of the trials.

6.3 Data Validity

The data analyzed herein have one major strength and one major weakness. Their weakness is that the data includes only straight-line paths. Although we tested these methods against a single curved path (Figures 3 and 6), errors may still exist in the approximation methods. However, since it includes angular velocity, we believe that the model itself is sound for curved paths.

The data's strength is their wide range of step frequencies. Our experience indicates that 1.0Hz (very slow) and 2.5Hz (very fast) are difficult step frequencies to maintain. We expect most step frequencies to remain within these ranges.

7 Discussion

Method-Recognized Walking: We examined why 32 of 246 trials had imperfect walking recognition scores. The principal problem is too many bobbing cycles in the window. In 20 of the imperfect-score trials, the subject walked at over 2Hz. As stated in Subsection 3.1, the window size was chosen to include 1.2 head-bob sinusoidal cycles at the lowest allowed step frequency (0.8Hz). At step frequencies of 2Hz or higher, the window contains at least 3 rightward-bob periods' data. The more cycles of walking data, the more likely some of the Forward Walking Model (Section 2) is violated. The two most common model violations are within-window step frequency variation, and within-window bobbing-free vertical head position ($c_{\mathbf{z}}$) variation.

The error in response to over-large window size partially explains why the Expedited Method trended toward better recognition scores: The model assumptions hold better because it uses a smaller window size. We believe a fruitful later enhancement will be to automatically adapt the window size based on step-frequency estimates.

Estimated Parameters: As Table 2 shows, the worst results are outliers in the Full-Model-Fit's speed with imperfect walking-recognition scores. There are two such trials. In both cases, the method found a local minimum with a poor estimate of angular velocity. An angular-velocity-initializing value (similar to those for walk speed, walk direction, and step frequency) would fix this problem.

Real-Time Considerations: Both approximation methods execute at real-time or better rates. However, we suggest the following areas for real-time-processing consideration:

- *Head turns*: When the head turns, the head's position moves somewhat – obscuring bobbing components. To minimize such effects, we suggest fitting the model to the position of base of the skull by rigid-body transform from the tracker location.
- *Latency*: The Full-Model-Fitting Method estimates parameters for a central value on a 3 second window (1.5s latency). The Expedited Method estimates Stage 1 parameters at the central value on a 1.5 second window (0.75s latency) and Stage 2 parameters at the central value on a 2 step-period window (˜0.4-1.0s latency). In steady-state real walking, the rate of change for step frequency, step length, etc. is low. Therefore, although these parameters are most accurate for the window's central time, systems can use them as the current value (zeroeth-order approximation; as done by Interrante et al. [6]). Higher-order approximations could lead to better latency-removed parameter estimates.
- *Trusting parameters*: Some parameters are more trustworthy than others. For instance, the estimates of amplitude and phase of the upward bob were usually both reliable and consistent. The forward bob was far less so (Figure 5). Any use of forward bob should be sensitive to the estimation errors the methods report.
- *Post-hoc filtering*: As is visible in Figure $3(b)$–(e), there are times that the walk-recognized state toggles. We have seen this most during starting or stopping periods. Similarly, when the system eliminates accumulated error after several steady-state calculations, some parameter's values show small discontinuities. We recommend using post-method filtering to remove both.

Asymmetric gait: We found asymmetric gait in non-pathological users – walking somewhat faster over one leg than the other. Our early data indicates that non-pathological asymmetric gait directly affects step length, but does not significantly affect step frequency. If this proves generally true, a one-half-step-frequency sinusoid should be added to the model's forward component.

Further analysis: We suggested that these estimated parameters could measurably improve three VE applications: animated self avatars, Redirected Walking, and per-user calibrated WIP systems. Further research must incorporate this work into such systems, and perform user studies to test this hypothesis. We have already begun such work.

Even without any of these further enhancements, we have demonstrated that important walking parameters can be reliably extracted from head-track data alone. Our validation demonstrated that our two approximation methods extract accurate step frequency, bobbing-free walk direction, and bobbing-free speed.

References

[1] J. Chai and J. K. Hodgins. Performance animation from low-dimensional control signals. In *Proceedings of ACM SIGGRAPH 2005*, volume 24, pages 686–695, July 2005.

[2] GAITRite. GAITRite systems. http://www.gaitrite.com/Products/index.html, March 2010.

[3] S. Hillaire, A. Lecuyer, R. Cozot, and G. Casiez. Using an eye-tracking system to improve camera motions and depth-of-field blur effects in virtual environments. In *Proceedings of the IEEE Virtual Reality, 2008*, 2008.

[4] V. T. Inman, H. J. Ralston, and F. Todd. *Human walking*. Williams and Wilkins, 1981.

[5] Innovative Sports Training. Motionmonitor version 8. http://www.innsport.com/, January 2010. Retrieved January 2010.

[6] V. Interrante, B. Ries, and L. Anderson. Seven league boots: A new metaphor for augmented locomotion through moderately large scale immersive virtual environments. In *IEEE Symposium of 3D User Interfaces 2007*, pages 167–170, 2007.

[7] A. Lecuyer, J.-M. Burkhardt, J.-M. Henaff, and S. Donikian. Camera motions improve the sensation of walking in virtual environments. In *Proceedings of the IEEE Virtual Reality 2006*, pages 11–17, 2006.

[8] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *The Quarterly of Applied Mathematics*, 2(2):164–168, 1944.

[9] G. Liu, J. Zhang, W. Wang, and L. McMillan. Human motion estimation from a reduced marker set. In *Proceedings of the ACM 2006 Symposium on Interactive 3D Graphics and Games*, pages 35–42, 2006.

[10] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11:431–441, 1963. Wikipedia condenses Levenberg-Marquard optimization concisely.

[11] S. Razzaque. *Redirected Walking*. PhD thesis, Department of Computer Science, The University of North Carolina at Chapel Hill, http://www.cs.unc.edu/Publications/PHDAbstracts.html#Razzaque, 2005. Retrieved May 2006.

[12] J. Rose and J. G. Gamble, editors. *Human Walking (3rd ed.)*. Lippincott, Wlliams & Wilkins, 2006.

[13] R. A. Ruddle and S. Lessels. The benefits of using a walking interface to navigate virtual environments. *ACM Transactions on Computer-Human Interaction*, 16(1):5:1–5:18, April 2009.

[14] M. Slater, M. Usoh, and A. Steed. Taking steps: The influence of a walking technique on presence in virtual reality. *ACM Trans. on Computer-Human Interaction*, 2(3):201–219, 1995.

[15] L. Terziman, A. Lecuyer, S. Hillaire, and J. M. Wiener. Can camera motions improve the perception of traveled distance in virtual environments. In *Procedings of the IEEE Virtual Reality 2009*, pages 131–134, 2009.

[16] M. Usoh, K. Arthur, M. C. Whitton, R. Bastos, A. Steed, M. Slater, and F. P. Brooks, Jr. Walking > walking-in-place > flying, in virtual environments. In *Proceedings of the ACM SIGGRAPH 1999*, pages 359–364, 1999.

[17] J. D. Wendt, M. C. Whitton, and F. P. Brooks, Jr. GUD WIP: Gait-Understanding-Driven Walking-In-Place. In *IEEE Virtual Reality*, pages 51–58, 2010.

[18] M. C. Whitton, J. V. Cohn, J. Feasel, P. Zimmons, S. Razzaque, S. J. Poulton, B. McLeod, and F. P. Brooks, Jr. Comparing VE locomotion interfaces. In *Proceedings of the IEEE Virtual Reality 2005*, pages 123–130, March 2005.

[19] J. Wuttke. lmfit – a C/C++ routine for Levenberg-Marquardt minimization with wrapper for least-squares curve fitting, based on work by B. S. Garbow, K. E. Hillstrom, J. J. More, and S. Moshier. version 2.5. http://www.messen-und-deuten.de/lmfit/, September 2009. Retrieved September 2009.