12-2011

Measuring Digital System Latency from Sensing to Actuation at Continuous 1 Millisecond Resolution

Weixin Wu
Clemson University, laviewwx@gmail.com

Measuring Digital System Latency from Sensing to Actuation at Continuous 1 Millisecond Resolution

---

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Electrical and Computer Engineering

---

by
Weixin Wu
December 2011

Accepted by:
Dr. Adam Hoover, Committee Chair
Dr. Stan Birchfield
Dr. John Gowdy

Abstract

This thesis describes a new method for measuring the end-to-end latency between sensing and actuation in a digital computing system. Compared to previous work, which generally measures the latency at 16-33 ms intervals or at discrete events separated by hundreds of ms, our new method measures the latency continuously at 1 millisecond resolution. This allows for the observation of variations in latency over sub 1 s periods, instead of relying upon averages of measurements. We have applied our method to two systems, the first using a camera for sensing and an LCD monitor for actuation, and the second using an orientation sensor for sensing and a motor for actuation. Our results show two interesting findings. First, a cyclical variation in latency can be seen based upon the relative rates of the sensor and actuator clocks and buffer times; for the components we tested the variation was in the range of 15-50 Hz with a magnitude of 10-20 ms. Second, orientation sensor error can look like a variation in latency; for the sensor we tested the variation was in the range of 0.5-1.0 Hz with a magnitude of 20-100 ms. Both of these findings have implications for robotics and virtual reality systems. In particular, it is possible that the variation in apparent latency caused by orientation sensor error may have some relation to "simulator sickness".

Acknowledgments

I would like to thank my advisor Dr. Hoover, for his guidance and inspiration throughout my research, and Dr. Muth for all his support to our project. I want to thank my committee members, Dr. Birchfield and Dr. Gowdy for their suggestions in the thesis. My thanks also belong to my caring parents and my loving wife. Without their support I wouldn't be able to come this far. I would also like to thank my colleague Yujie for his help in my research. And to all my friends, thank you all.

Table of Contents

Chapter 1

Background

Latency is a problem. This thesis considers the problem of measuring input-to-output latency in a computing system. The system takes some amount of time to sense the signal, process the signal, and output the result. Even if the signal is slightly changed, such as for an object in front of a camera merely forwarded to the display, it takes some time to move across the various memory buffers and buses in the computing system. The same happens to a orientation sensor, if the orientation reading changes a little, such reading need to go through all the integration process, various buffers and drivers before the output. System latency is a major concern in several applications, particularly in virtual environments and in robotics.

In a virtual environment, the movements of a user are tracked and used to drive a head mounted display (HMD). The images seen by the user are supposed to reflect the motions the user is undergoing in the real world. If the system latency is too large, then the virtual experience will not be perceived as real. The user will feel that the display is lagging, and may become nauseous. In robotics, cameras are often used to drive robot motion. For example, a camera can be used on a mobile robot to navigate around obstacles. A camera can also be used on a robot arm, to guide the capturing and releasing of objects. If the system latency is not accounted for in a robot, then it runs the risk of colliding with moving obstacles, or of failing to grasp moving objects.

In order for these systems to operate correctly, they must account for the system latency. Typically, latency is on the order of tens to hundreds of milliseconds. The most common approach is to assume that the latency is constant, and to use that constant to make predictions about the desired system output. For example, if a ball is thrown to a robotic arm, the arm can catch the

ball by projecting its trajectory into the future according to the system latency. In general, in a virtual environment, a user typically only moves smoothly in short bursts, a mobile robot navigating around humans cannot assume that the humans will always continue along tracked trajectories. The second problem is that the system latency is not constant. Latency varies depending on a variety of factors, because of the asynchronous nature of the parts working together in the system. In fact, the distribution of latency is not Gaussian, so an average and standard deviation are not appropriate as a measure. For these sorts of systems to operate precisely, it is likely that a more advanced model of system latency needs to be used.

A precise measurement enables methods designed to compensate for the latency. Precise measurements also are necessary for research into architectures and methods intended to reduce latency. So we are interested in measuring, to an accuracy of 1 ms, how long this entire process takes. We are particular interested in the system latency distribution and its variance.

Chapter 2

Introduction

This thesis considers the problem of measuring the latency in a digital system from sensing to actuation. We are motivated by sensors and actuators that operate using their own clocks, such as digital cameras, orientation sensors, displays and motors. Figure 2.1 shows a typical configuration. The system latency, also called end-to-end latency, is defined as the time it takes for a real-world event to be sensed, processed and actuated (e.g. displayed). Latency is commonly in the range of tens to hundreds of ms, and thus while difficult to measure, is in the range that affects control problems and human end users. In virtual reality systems, latency has been shown to confound pointing and object motion tasks [17], catching tasks [7] and ball bouncing tasks [13]. In robotics, latency has an impact on teleoperation [19] and vision-based control [8]. Its effect has also been studied in immersive video conferencing [14].

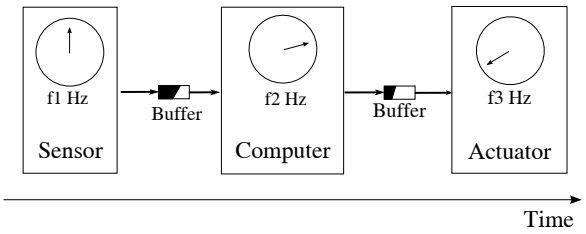


Figure 2.1: System latency is non-constant due to components using independent clocks and the variable delays in buffers connecting components.

It is possible to measure latency internally using the computer in the system, by time-stamping when a sensor input is received, and by time-stamping when an actuation output is com-

manded. However, these time-stamps do not include the time that data may spend in buffers, nor do they include the time that may be spent by the sensor acquiring the data or by the actuator outputting the data. Therefore it is preferable to use external instrumentation to measure the latency by observing the entire system. Two general approaches have been taken to this problem, one that uses a camera to continuously observe the system, and one that uses event-driven instrumentation such as photodiodes to more precisely measure discrete events.

Figure 2.2 illustrates a typical experimental setup for the camera-based continuous approach. A sensor (usually a component of a 3 DOF or 6 DOF tracking system) is placed on a pendulum or other moving apparatus. A computer receives the tracking data from the sensor and displays it on a monitor. An external camera views both the live motion and the displayed motion, comparing them to determine the latency. Bryson and Fisher [3] pioneered this approach by comparing human hand movement of a tracked device against the displayed motion; latency was calculated as the number of camera frames between when hand motion started and when the displayed motion started. He et al. [5] used a similar approach with a grid visible behind the tracked object so that multiple points could be used for measurements. Liang et al. [6] was the first to suggest using a pendulum to move the sensor so that the actual motion was known; latency was calculated as the time between when the camera frames showed the pendulum at its lowest point versus when the tracked data showed the pendulum at its lowest point. Ware and Balakrishan [19] followed the same approach but used a motor pulling an object back and forth linearly so that the tracked object velocity was constant. Steed [15] also used a pendulum but fit sinusoidal curves to both the live and displayed data, calculating the relative phase shift between the curves, so that a more precise estimate of latency could be made. In one experiment, Morice et al. [13] used a racket waved in an oscillatory motion by a human; latency was measured by finding the time difference between frames containing the maxima of the motion in the live and displayed data. Swindells et al. [16] used a turntable; latency was measured using the angular difference between the live and displayed data. Instead of using a camera to observe the system, Adelstein et al. [1] moved the tracked object using a robot arm; latency was measured by comparing the angle of the motor encoder of the arm against the angle of the tracking sensor. All of these methods are capable of measuring latency continuously, but the reported experiments were limited by the sampling rates of the cameras or instrumentation (25-50 Hz). Because the measured latency is in the range of 30-150 ms, multiple measurements were averaged or data was interpolated in-between measurements. In contrast, because our method uses

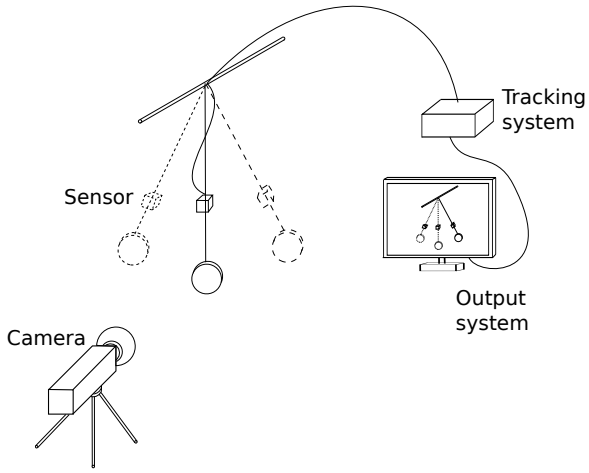


Figure 2.2: Continuous approach to measuring latency.

a 1,000 Hz camera, we can continuously measure latency at 1 ms intervals in order to see variations in the latency not discernible at slower resolutions.

Figure 2.3 illustrates a typical experimental setup for the discrete event-based approach to measuring latency. In this approach, a photodiode is placed at a fixed position so that when the tracked object passes that point a signal is registered on an oscilloscope. A second photodiode is placed at the corresponding fixed position for the displayed output. This approach was pioneered by Mine [12], who used several variations of the idea (with different instrumentation) to estimate latency in different parts of the systems of interest. The method has been used by other researchers with similar results [2, 13, 10, 17]. While this approach allows for more precise measurements of latency (because the instrumentation is not limited to the sampling rate of a camera), measurements can only be made at the discrete times when the tracked object passes the reference point. This approach does not account for variations in latency that may happen at different positions of the sensor and actuator; for example, actuation in a display monitor takes place at different times across the screen as the image is redrawn. All of the experiments reported using this approach calculated average latencies, and did not describe latency variation over time.

Miller and Bishop [11] describe a method to calculate latency continuously using 1D CCD arrays operated at 150 Hz. However, they average their calculations from these measurements in such a way that latency is only calculated at 10 Hz. DiLuca et al. [9] describe a method using photodiodes moved sinusoidally in front of a sensed and displayed gradient intensity. The variations in intensity

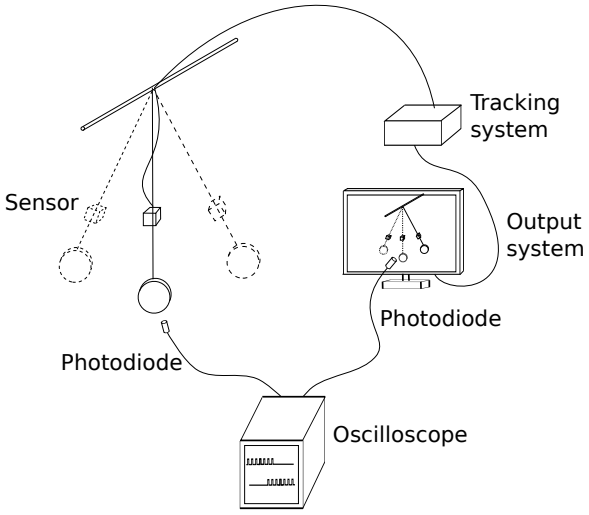


Figure 2.3: Discrete event approach to measuring latency.

are correlated to calculate the average latency. In their experiments they used a stereo input of a laptop computer, presumably operating at a 44 KHz frequency (this detail was not provided in the paper). However, the measurements were high-pass filtered and then correlated to find an average. Although their method potentially could be used to study continuous variations in latency, they did not pursue this idea.

All the works discussed above report average latencies. Our own previous work in robotics made the same assumption [8], estimating an average and standard deviation for latency, and compensating for manipulation tasks by building the gripper large enough to capture the majority of the distribution. Previous works have discussed the idea that system latency is not a constant [1, 9]. However, our method is the first to show how to continuously measure the latency at a rate sufficient to see how it changes over a period less than 1 second.

Chapter 3

Methods

Our approach is similar to other continuous methods discussed in the introduction. Figure 3.1 illustrates our methodology. The system being measured is configured in such a way that the actuator outputs the same property (e.g. position, angle, etc.) sensed by the sensor. The "outside observer" (we use this term to differentiate it from any camera used as a sensor in a system being measured) is a high-speed camera capable of observing the property. Latency is measured by calculating the number of high-speed camera frames between when the sensed property matches the actuated property. We performed experiments on two systems using this approach. We first describe our outside observer, then describe each system in detail.

3.1 Outside observer

For an outside observer we used a Fastec Trouble Shooter 1000 high speed camera. It can capture video at 480 × 640 resolution at up to 1,000 Hz for 4.4 seconds. We have found that at

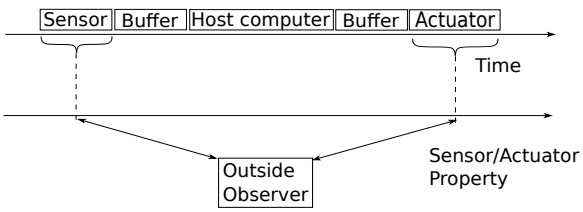


Figure 3.1: Latency is measured indirectly via the property (e.g. position, orientation) being sensed and actuated.

this speed, the scene being imaged must be very brightly illuminated, because the exposure interval is so small. To compensate we use external spotlights mounted around the systems to increase the ambient illumination. Because the spotlights operate at 60 Hz synchronous to the power source, they cause an oscillation in intensity in the high speed camera frames. To address this problem, adaptive thresholding (discussed later) is used during the processing of the images.

3.2 System #1

Our first system uses a camera for sensing and a computer monitor for actuation. The camera is a Sony XC-75 (`www.subtechnique.com/sony/PDFs/xc-7573e.pdf`), an interlaced camera operating at 30 Hz. The computer has an Intel Core Duo 2.8 GHz processor, 4 GB main memory and a 500 GB hard drive. The frame grabber is a Matrox Meteor-II Multi Channel (`http://www.matrox.com/imaging/en/products/frame_grabbers/`). The graphics card is a nVidia Geforece 9500 GT (`http://www.nvidia.com/object/product_geforce_9500gt_us.html/`). The operating system is Windows XP Professional SP2. The monitor is an Acer AL2216W operating at 60 Hz.

Figure 3.2 shows a diagram of the experimental setup. The sensor is aimed at a specially constructed apparatus, labeled the "sensed input event" in Figure 3.3. The images captured by the sensor are digitized in the computer and forwarded to the actuator, an LCD display. The computer does not change the content of the sensed images, so that the output image matches the sensed input image, but after some latency. The outside observer sits behind the system with its field-of-view positioned so that it can see the sensed input event and the actuated output event simultaneously. By comparing these and matching when they show the same content, we can indirectly measure the latency.

Figure 3.3 shows a picture of the apparatus. It consists of a background piece of wood painted white, with a wooden bar painted black in front of it. The bar is fixed vertically so that it can only move back and forth horizontally. The purpose of the apparatus is to create a motion that is easily discernible in the high speed captured images. This facilitates image processing of the frames captured by the outside observer, in order to help automate the measurement process. During an experiment, the black vertical bar of the apparatus is manually moved horizontally.

An example raw frame captured by the outside observer is shown in Figure 3.4. The sensed

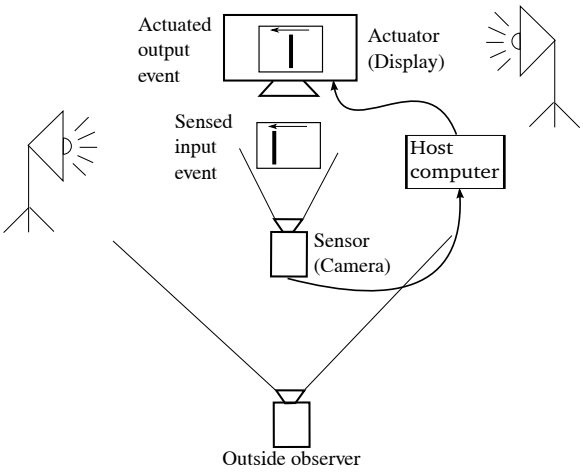


Figure 3.2: System #1: camera to monitor.

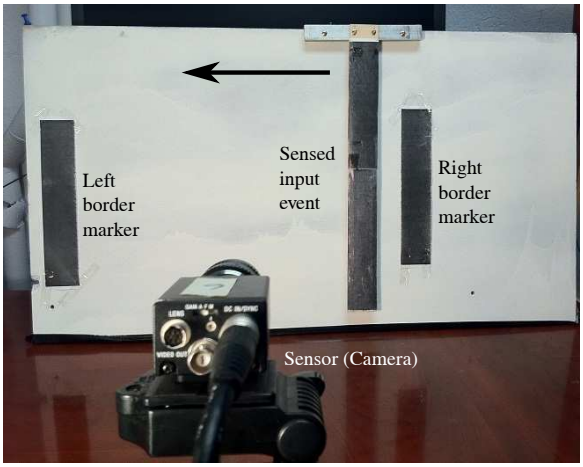


Figure 3.3: Camera-to-monitor experiment apparatus.

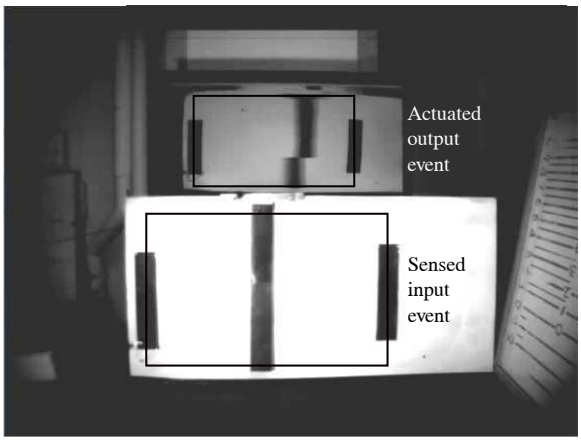


Figure 3.4: Camera-to-monitor system as seen by the outside observer.

input event is visible in the lower section and the actuated output event is visible in the upper section. The latency can be seen by the different positions of the bar. The tear in the bar in the actuated output is due to the redrawing of the image in the LCD monitor. This is discussed more in the results.

Automated image processing is used to take measurements from the raw frames captured by the outside observer. The processing only happens within the windows highlighted in Figure 3.4. The steps of the processing include histogram equalization, adaptive segmentation and binarization. The histogram equalization compensates the exposure to a human-visible level and reduces the variation of intensity between frames, which leads to cleaner object segmentations. In the adaptive segmentation process, a threshold based on the histogram is computed and used to segment the object of interest. In the binarization process, the grayscale image is converted to a binary image, where a pixel value of 0 indicated background a value of 1 indicates object. An example segmented frame is shown in Figure 3.5.

3.2.1 Sensing and actuation property

For system #1, we define the sensed and actuated property as the position of the black vertical bar as a percentage of its distance from the right border marker to the left border marker (see Figure 3.3). We used percentage rather than raw position to simplify calculations that determine when the actuated output event is in the same position as the sensed input event. The horizontal positions of the border markers were manually marked as shown by L and R in Figure 3.5. The top

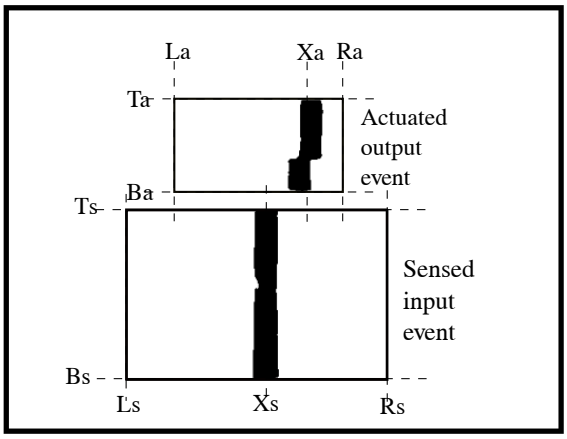


Figure 3.5: Property (position) measured by outside observer.

T and bottom B boundaries of the areas of interest were also manually marked. Note that these only needed to be marked once during experimental setup, because the boundaries did not move during experiments.

The position of the sensed input event is calculated as the object's 1st order moment in x-coordinates:

$$X_s = \sum_{y=B_s}^{T_s} \sum_{x=L_s}^{R_s} x^p y^q I(x,y) \tag{3.1}$$

where p is 1, q is 0, $I(x,y)$ is the segmented binary image, and X_s is the sensed input event's position. The sensed input property (position percentage) is then computed as:

$$P_s = \frac{\|X_s - L_s\|}{\|R_s - L_s\|} \tag{3.2}$$

The position of the actuated output event is calculated similarly, substituting a for s subscripts for the variables shown in Figure 3.5 into Equations 3.1 and 3.2.

3.2.2 Mapping property to latency measurements

For each outside observer frame we measure P_s and P_a. These can be plotted over time (over consecutive outside observer frames) as shown in Figure 3.6. To measure the latency at a particular frame P, we find the frame P' where the actuated output property P'_a is equal to the

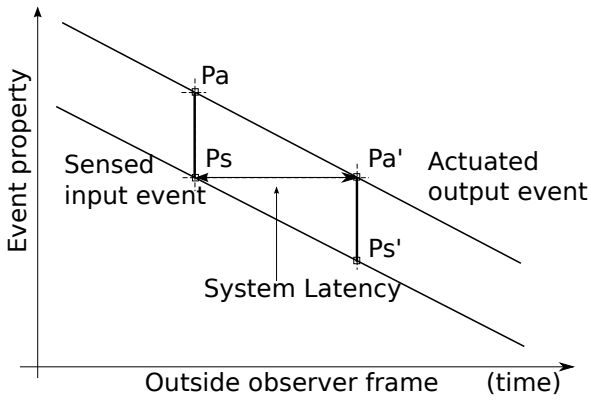


Figure 3.6: Mapping property measurements to latency measurements.

sensed input property P_s. This latency can be computed independently for every outside observer frame.

3.2.3 Modeling the camera-to-monitor system latency

In this section, we briefly discuss a timing model of the expected latency in system #1. We used this model to generate simulated histograms of the latency, depending upon the settings of the camera and monitor. For example, we can change the shutter speed of the camera and the refresh rate of the monitor. We used this model to compare our measurements of the actual system against the histograms generated by our simulation.

The simulation model is based upon events, and uses 5 parameters to control the flow of information from sensing through actuation. The parameters are (1) the time data is being sensed, (2) the sensor clock rate, (3) the actuator clock rate, (4) the time data is being actuated, and (5) the total time data is being processed by the computer. For system #1, these parameters correspond to the CCD exposure time, the CCD frame rate, the LCD refresh rate, the LCD response time, and the computer processing time. The clock rates were set equal to those of the real components. The total time spent in processing was determined by internal measurement within the program that processes the data; specifically, timestamps at the acquisition of data and the output of data were differenced and averaged over multiple runs. The times spent in sensing and actuation were arrived at through a combination of theoretical modeling about how the components work as well as measurements using the high speed camera. The simulation runs by propagating an event, in the case of system #1 an image, from sensing all the way through actuation. The end-to-end latency is

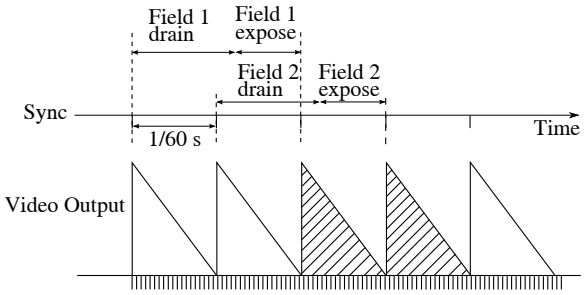


Figure 3.7: CCD camera imaging time line.

determined as the time between the mid-point of sensing (the average of the accumulation of image charge) to the mid-point of actuation.

As for the system #1, the sensing process can be decomposed into fields and frames. In our experiment both fields were used to generate one frame. Two of the fields are combined into one frame. The time line of the sensing process is shown in the Figure 3.7. The propagation latency can be computed using the Equation 3.3. The t_{field} and t_{expo} are the two parameters that describe the exposure process. In our experimental camera, the t_{field} is fixed time amount of 33 ms. The t_{expo} is the variable exposure time, which ranges from 1/2000 to 1/60s.

$$latency_{camera} = \frac{3}{2}t_{field} + \frac{1}{2}t_{expo} \tag{3.3}$$

The Table 3.1 lists the major latency components. Note these latency components are not independent.

Process	Clock Rate/Bandwidth	Latency
CCD imaging	30 Hz	35 - 45ms
FG to MM transfer	160 MB/s	2 ms
CPU processing	$>$ 1 GHz	$<$ 1 ms
MM to GC transfer	4 GB/s	$<$ 1ms
LCD polling	60 Hz	16.67 ms
LCD unit ignition		5 - 10 ms

Table 3.1: Processes involved in camera-to-monitor system.

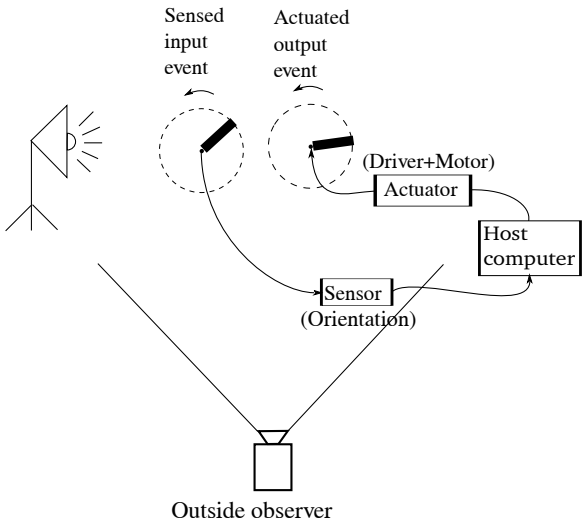


Figure 3.8: System #2: orientation sensor to motor.

3.3 System #2

Our second system uses an orientation sensor for sensing and a motor for actuation. The sensor is an InertiaCube (`www.intersense.com/pages/18/11/`); it uses the filtered results of 3-axis gyroscopes, magnetometers and accelerometers to determine 3DOF angular pose at 110 Hz. The computer configuration is the same as in system #1. The motor is a Shinano Kenshi SST55D2C040 stepper motor (`www.pikpower.com/New%20Site/New_pdfs/SKC/SKCnew.pdf`). The motor driver is an Applied Motion Si2035 (`www.applied-motion.com/products/stepper-drives/si2035`).

Figure 3.8 shows a diagram of the experimental setup. The sensor is mounted on an apparatus that can be manually rotated. The computer reads the sensor and turns the motor to the same orientation. The outside observer is positioned so that it can view both orientations. By comparing the two orientations, we can indirectly measure the system latency.

Figure 3.9 shows an example image captured by the outside observer. The sensor is mounted on a black bar that emphasizes one of the three angles of the orientation sensor. The actuator is similarly mounted with an bar attached to it so that its rotation can also be viewed by the outside observer.

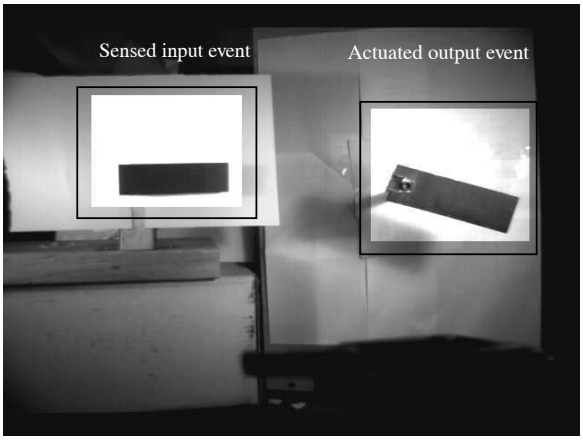


Figure 3.9: Orientation-to-motor system as seen by the outside observer.

3.3.1 Sensing and actuation property

For system #2 we define the property of interest as the direction of the black bar in the local coordinate system of both the sensed input event and the actuated output event. At startup, we assume the bars point in different directions and so define the initial orientation of each as 0° in its local coordinate system. We use automated image processing to determine the direction. Equalization, adaptive thresholding, and segmentation are carried out as described previously. Figure 3.10 shows an example result after adaptive thresholding and segmentation. The angle is computed by calculating a local eigenvector for each segmented object using moments and central moments. The pth and qth moments are computed as:

$$m_{pq} = \sum_{x} \sum_{y} x^p y^q I(x, y) \tag{3.4}$$

The center of the object is computed as:

$$(x_c, y_c) = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right) \tag{3.5}$$

The central moments are computed as:

$$\mu_{pq} = \sum_{x} \sum_{y} (x - x_c)^p (y - y_c)^q I(x, y) \tag{3.6}$$

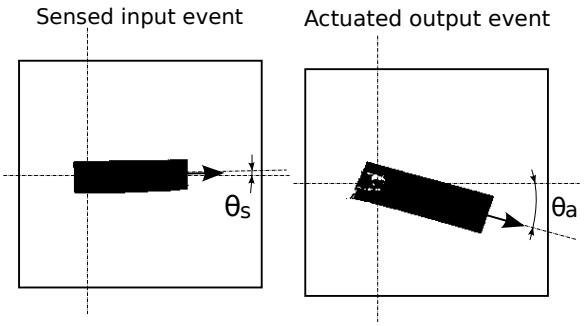


Figure 3.10: Property (orientation) measured by outside observer.

Finally, the direction is computed as:

$$\theta = frac12atan2\left(2\mu_{11}, \mu_{20} - \mu_{02}\right) \tag{3.7}$$

where θ denotes the direction. The last step is to compensate for the difference between the initial orientations of both bars. This is done by subtracting the angle computed from the outside observer's first frame for each bar.

For each outside observer frame we measure θ_s and θ_a. These can be plotted over time as shown previously in Figure 3.6. Latency can then be calculated as described previously.

Chapter 4

Results

4.1 System #1

Figure 4.1 shows the result for measuring latency continuously for system #1 over a 700 ms period of time. Comparing this result to Figure 3.6 shows that the latency is not constant (both lines are not straight). Instead, the latency is varying by approximately 17 ms at a 33 ms frequency. This represents the interplay between the 30 Hz clock of the sensor (camera) and the 60 Hz clock of the actuator (monitor). The default exposure time for the camera is 33 ms, equal to its clock rate; therefore the snapshot of information captured in an image is an integral (or blur) across 33 ms. The default refresh time for the monitor is 17 ms, equal to its clock rate; therefore the actuation (or delivery) of its information takes place evenly across 17 ms. Figure 3.1 emphasizes this idea, that neither sensing nor actuation happens in an instant. Because the events happen at frequencies higher than 20 Hz, human observers perceive them as continuous. Our method for measuring latency shows how the latency actually looks at 1 ms resolution, as the amount of sensed data observed to have completed actuation varies.

Figure 4.2 shows how the latency for system #1 varies over time. If multiple measurements are randomly (or continuously) accumulated into a histogram, the distribution appears uniform. However, it is important to note that the end user of the system is not receiving actuated output from a random uniform distribution; the delay perceived by the end user follows a cyclical pattern.

Figure 4.3 shows the distribution of latency calculated from the data shown in Figure 4.1. It is not perfectly uniform because of noise during our measurement process (during image processing).

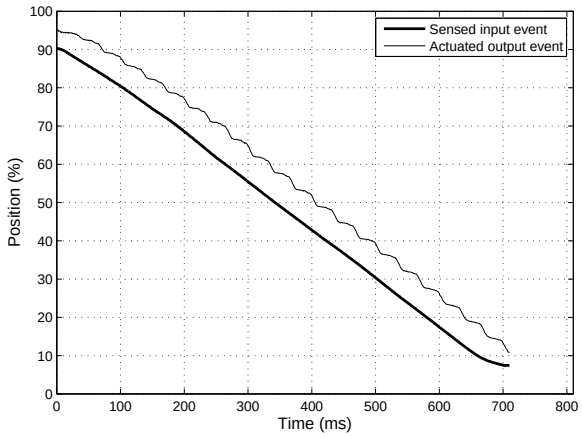


Figure 4.1: Measured sensed input and actuated output for system #1.

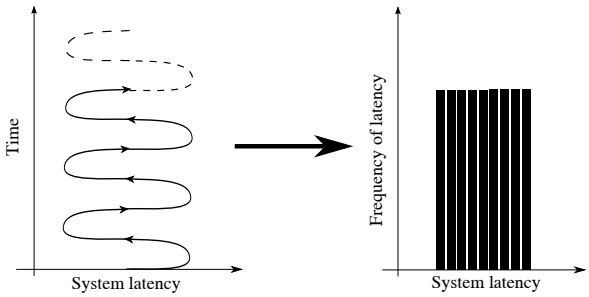


Figure 4.2: Latency perceived by end user of system #1.

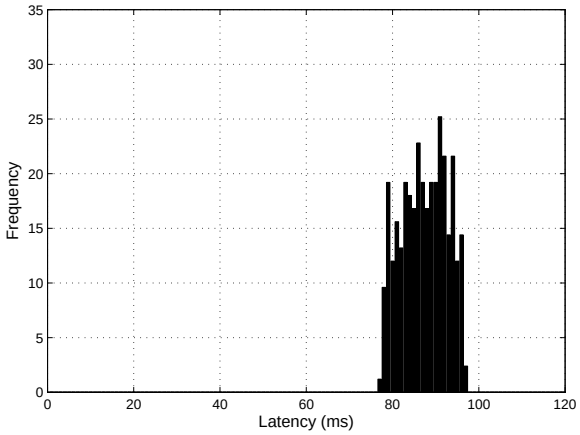


Figure 4.3: Distribution of latency measured for system #1.

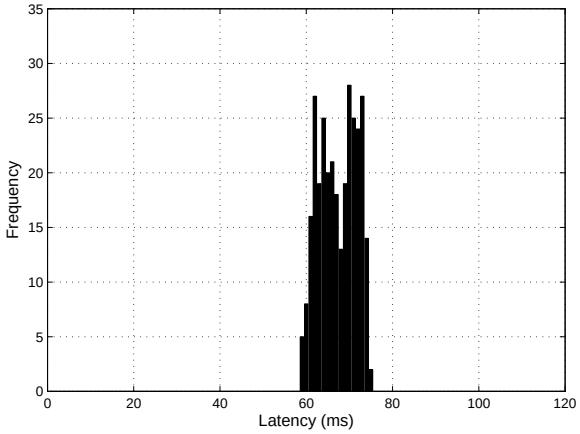


Figure 4.4: Distribution of latency measured for system #1, with sensor (camera) using a faster shutter speed.

For a second test of the same system, we changed the exposure time of the sensor (camera) from 33 ms to 2 ms. Note that this did not change the clock rate of the sensor, only the amount of time integrated into an image during sensing (see Figure 3.1). Therefore we expect an approximately 17 ms decrease in the distribution of latency. Figure 4.4 shows the result for measuring the distribution of latency for the faster shutter, confirming our expected decrease but otherwise showing the same shape.

As discussed previously, we created a model of system #1 in order to simulate measuring its latency and compare that against our real measurements. The only variables in the model are the clock rates of the sensor and actuator, and the amount of time spent in sensing, processing and

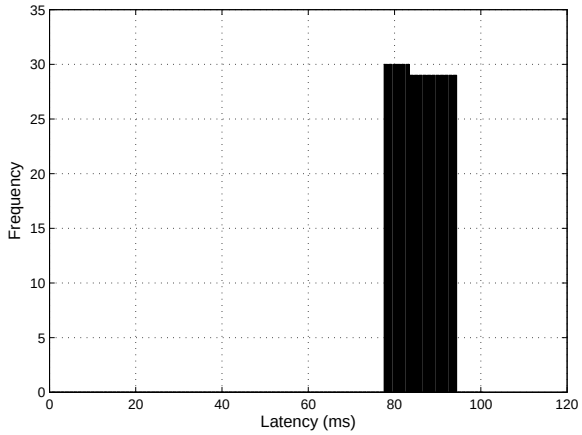


Figure 4.5: Simulated distribution of latency for system #1.

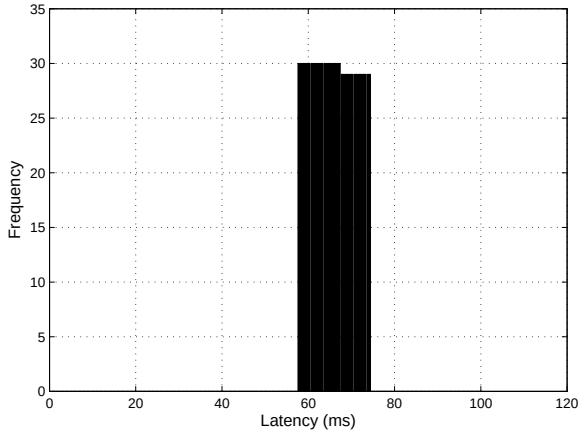


Figure 4.6: Simulated distribution of latency for system #1, with sensor (camera) using a faster shutter speed.

actuation. Figure 4.5 shows the result when the sensor (camera) has a 33 ms shutter speed, and Figure 4.6 shows the result when the sensor has a 2 ms shutter speed. Comparing these distributions to those shown in Figures 4.3-4.4 shows that they match against our measured results. This indicates that for purposes of modeling the latency, the necessary variables are the sensor and actuator clocks and the times spent in each of the three steps.

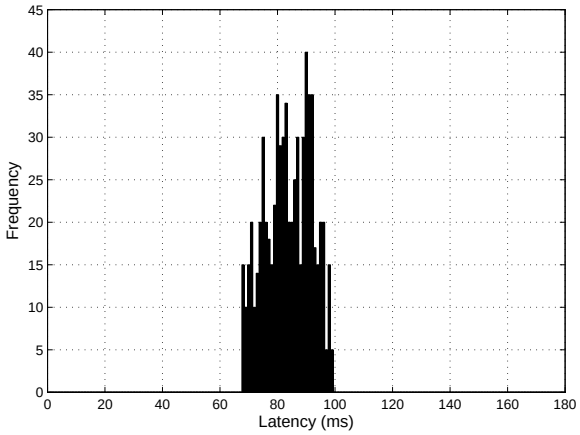


Figure 4.7: Distribution of latency measured for system #2, first trial.

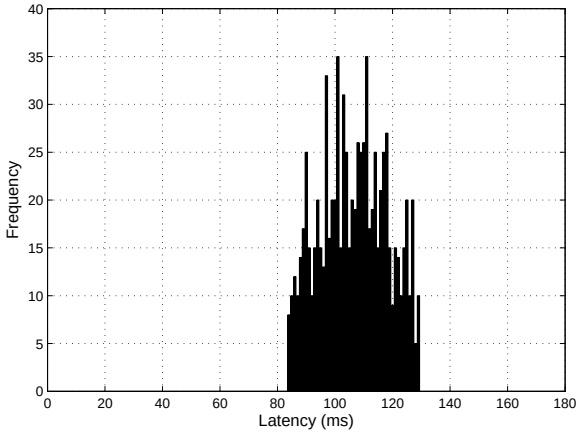


Figure 4.8: Distribution of latency measured for system #2, second trial.

4.2 System #2

The experiment for system #1 was repeated many times and always showed the same latency distribution. However, for system #2, the distribution changed between trials. Figure 4.7 shows the measured distribution of latency for one trial and Figure 4.8 shows the distribution for a second trial. Looking only at these plots, or similarly only calculating averages, it is uncertain what is causing the difference in measured latency. Using our method to plot the latency continuously at 1 ms resolution reveals more information.

Figure 4.9 shows the continuous measurement of the orientation property of both the sensed input and actuated output, for the first trial. First, note that the step-like shape of the actuated

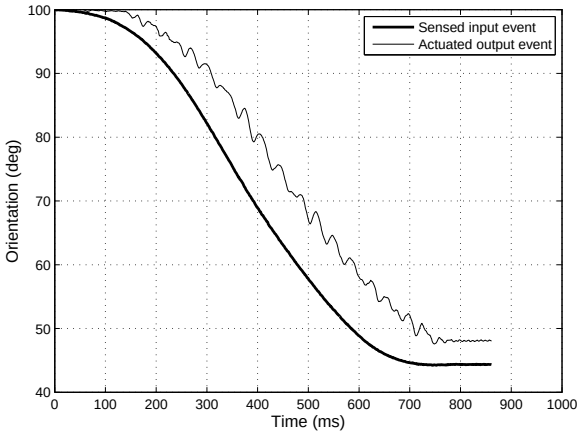


Figure 4.9: Measured sensed input and actuated output for system #2, first trial.

line is similar to that observed for system #1 (see Figure 4.1), showing the interplay of the sensor and actuator clocks. Second, note that the lines are not parallel. The angular difference between the orientation sensor and motor was artificially set to 0° at initialization, but drifted to 5° after 800 ms at the end of the trial, as the sensor was rotated through approximately 50°. This is consistent with the amount of error our group has observed in the angular reading provided by this sensor [18]. The result of this drift in sensor error is that the latency, which is the horizontal distance between the two lines, is slowly changing throughout the trial. It is important to note that this is not "real" latency, in-so-far as the system is not taking a differing amount of time to propagate the sensor readings through the system. However, the latency is *apparent* to the end user of the system because the time for the state of the output to match the state of the input is changing.

Figure 4.10 shows the same plot for the second trial. In this case, the sensor error was approximately −2° by the end of the 800 ms trial. Note again that the amount of horizontal distance between the two lines is varying.

More trails are listed in Figure 4.11 and 4.12.

In order to characterize this variation, we fit sinusoidal curves to the apparent latencies (the horizontal differences between the lines in Figure 4.9). Figure 4.13 and 4.14shows the raw measured latencies along with the fitted sine curve. The data were taken from the middle 400 ms of the trial where the calculation of latency is meaningful (at the beginning and end of the trials, when the object is not in motion, the latency cannot be determined). Note that the raw measurements are step-like because of the previously discussed interplay between the sensor and actuator clocks. The

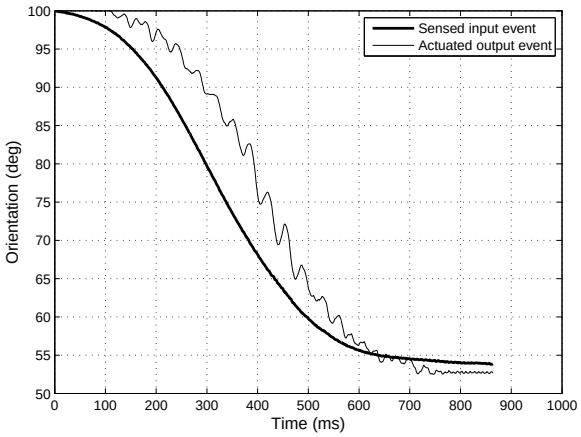
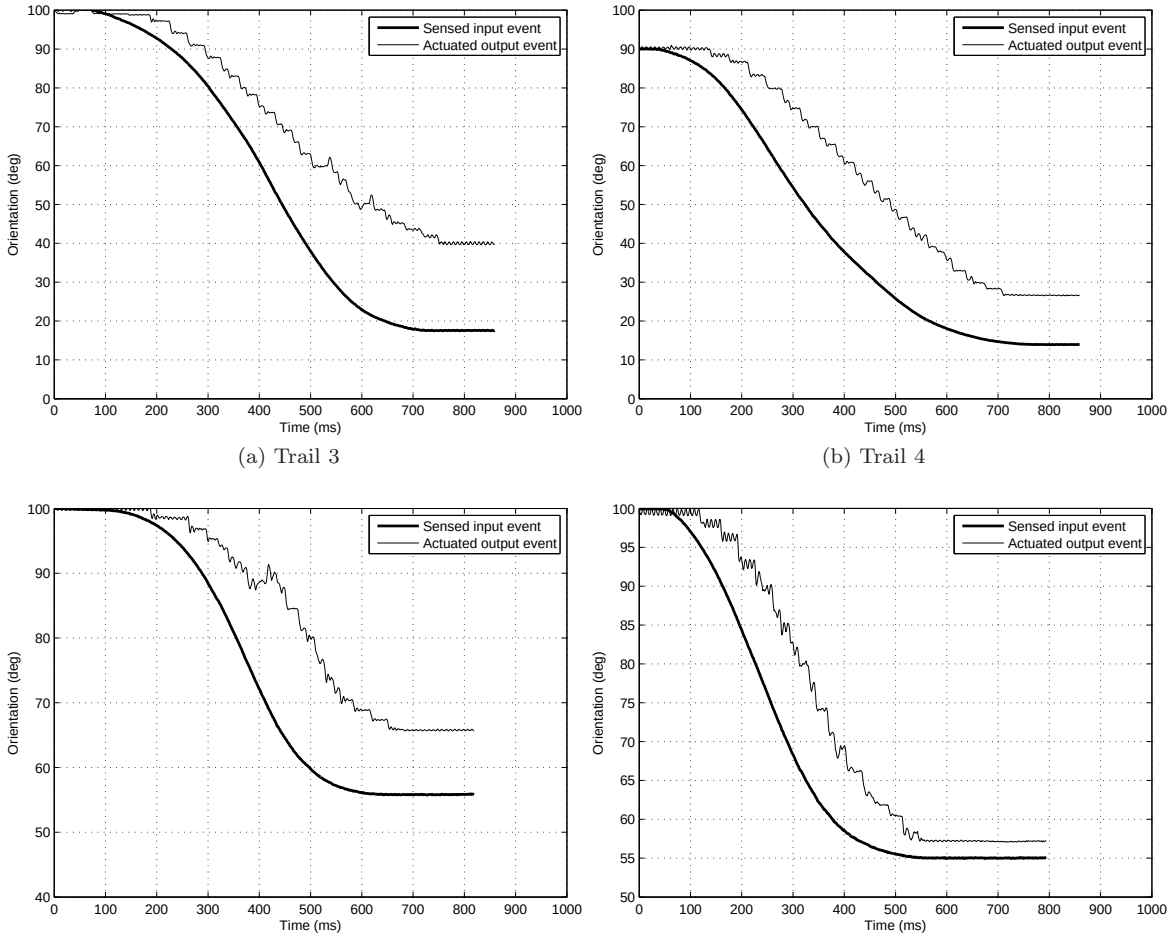


Figure 4.10: Measured sensed input and actuated output for system #2, second trial.



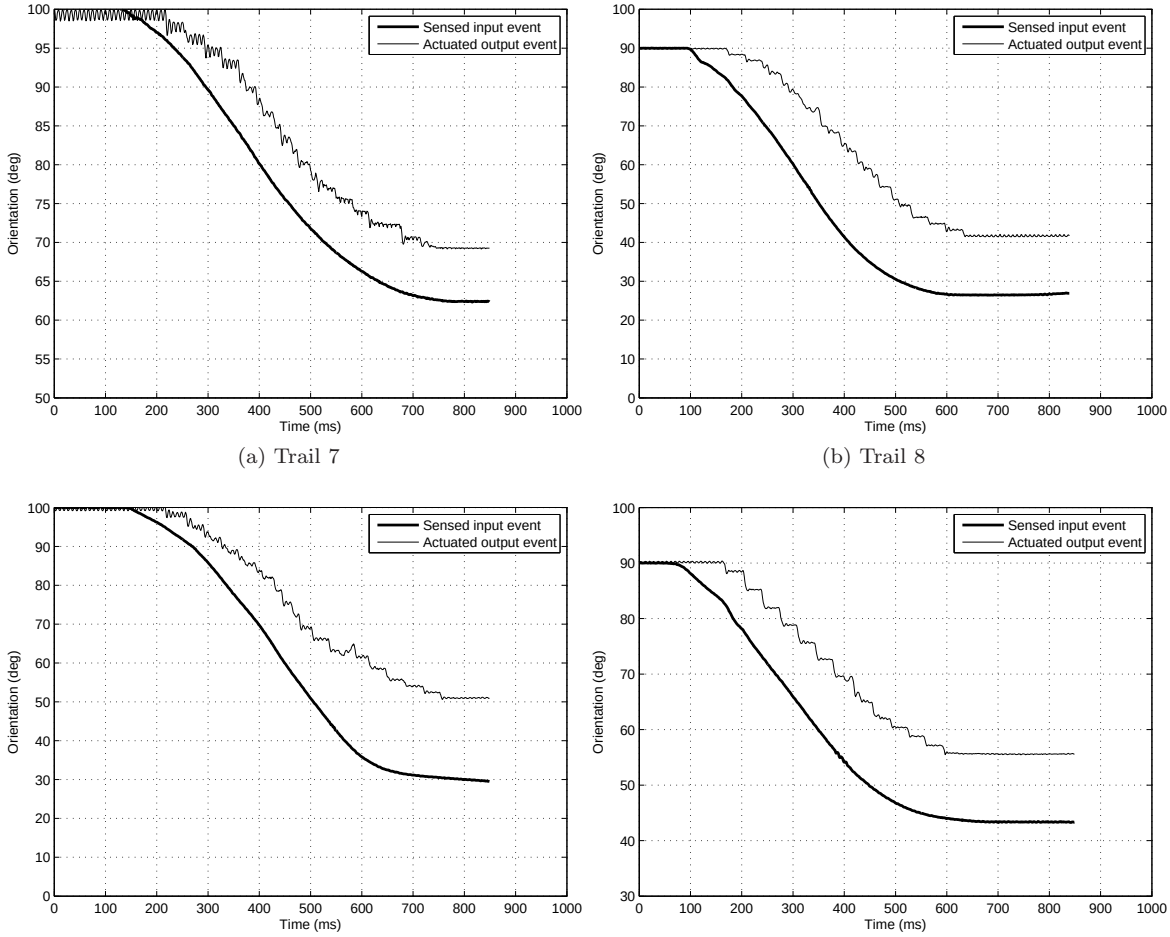
(a) Trail 3

(b) Trail 4

(c) Trail 5

(d) Trail 6

Figure 4.11: Measured sensed input and actuated output for system #2, cont.



(a) Trail 7

(b) Trail 8

(c) Trail 9

(d) Trail 10

Figure 4.12: Measured sensed input and actuated output for system #2, cont.

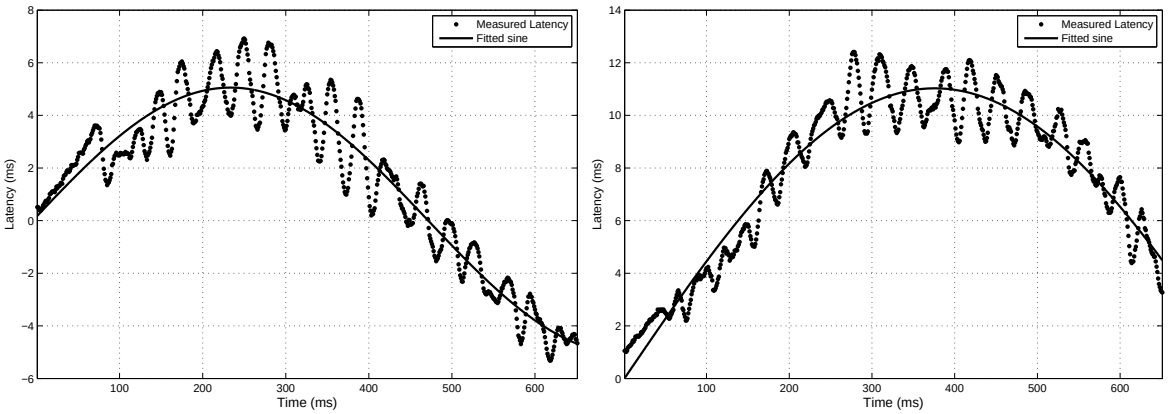
Latency (ms)	Frequency (Hz)
27.2	0.73
24.0	0.38
72.9	0.59
13.4	0.83
22.6	1.78
82.2	0.72
23.3	1.01
46.7	1.07
50.6	0.67
97.4	0.95

Table 4.1: Frequencies and magnitudes for apparent variations in latency, for ten trials with 50° rotational motion.

fitted sine shows the gradual change in latency as the sensor error drifts. From this figure it can be observed that the frequency of the perceived drift in latency is in the 0.5-1.0 Hz range, and that the magnitude of the perceived oscillation in latency is approximately 20-30 ms.

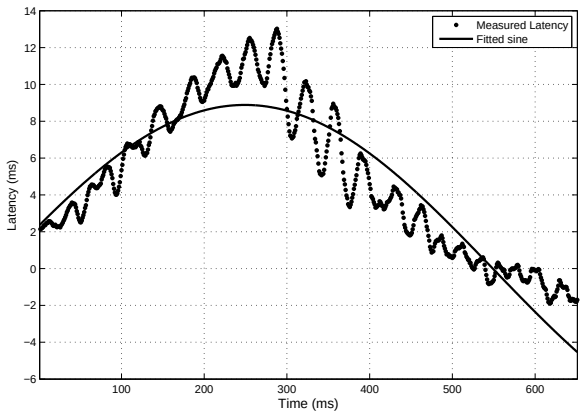
We repeated this process for ten trials. Table 4.1 lists the frequencies and magnitudes found for the fitted sines. Note that they vary due to differing amounts of sensor error in each trial, but the frequencies are generally in the 0.5-1.0 Hz range and the magnitudes are generally in the 20-100 ms range. This amount of apparent change in latency is certainly within the range perceivable by human end users. It is also well-known that frequencies in this range, such as those caused by ocean waves and vehicle motions, are among the worst for causing sickness in humans [4].

The motion in our first ten trials was approximately 50° of constant velocity rotation in 800 ms. For a human turning his or her head, this motion is not unreasonable, but it is relatively far. We repeated this test with a slower, shorter rotation of approximately 10° in 800 ms. We conducted 7 trials and fit sinusoidal curves to the apparent latencies. Figure 4.15 and 4.16 shows an example of raw measured latencies and fitted sine for one of the trials. Table 4.2 shows the calculated frequencies and magnitudes for the 7 trials. We found that they are in the same range as for the first set of tests. This implies that the sensor error is relatively independent of the speed of the motion, which matches our previous findings for evaluating the performance of the sensor [18]. It also shows that the sinusoidal variation in apparent latency, perceived by the user of a system incorporating this sensor, is independent of the speeds of motions made by the user.

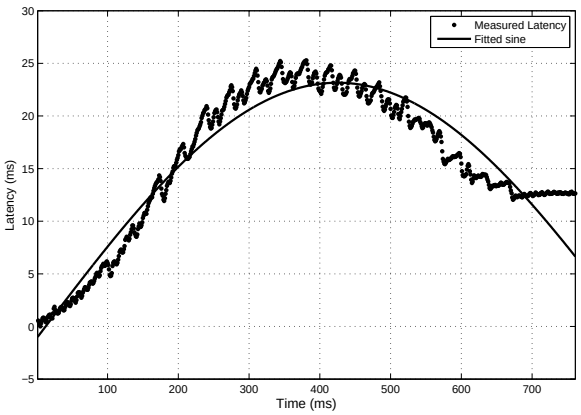


(a) Trail 1

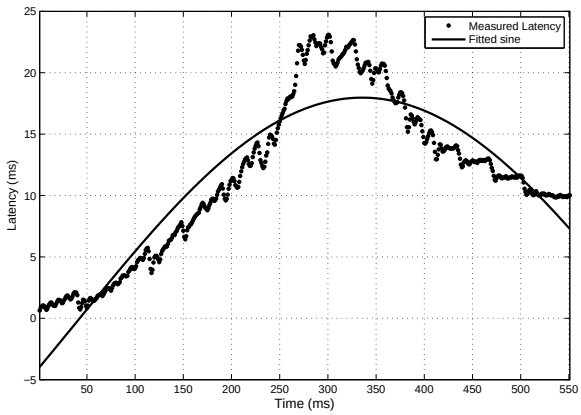
(b) Trail 2



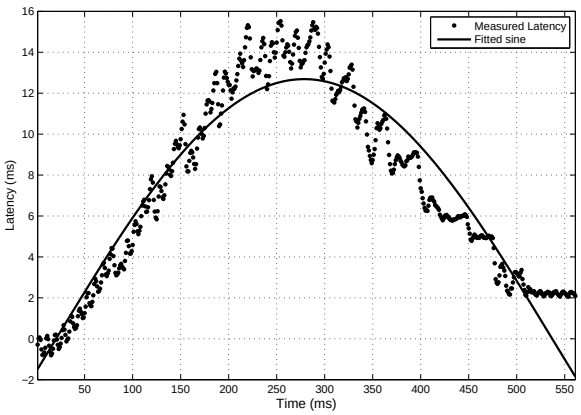
(c) Trail 3



(d) Trail 4

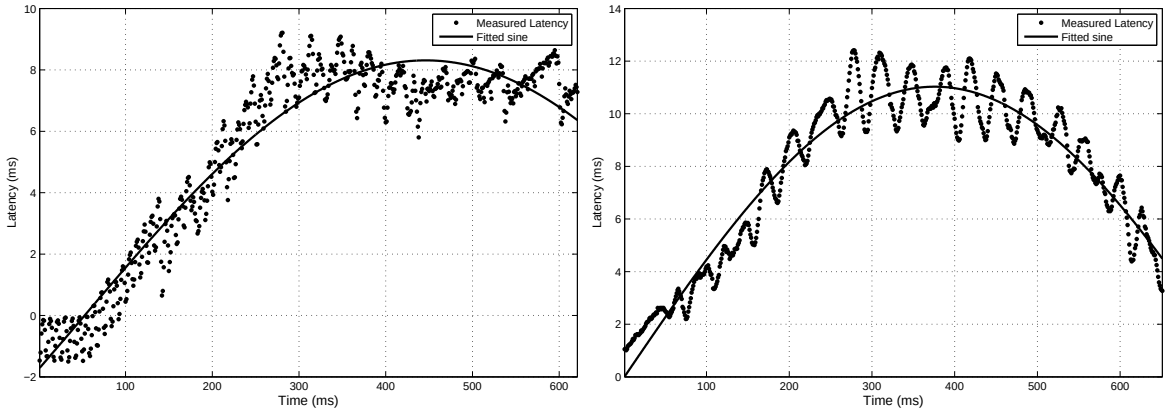


(e) Trail 5



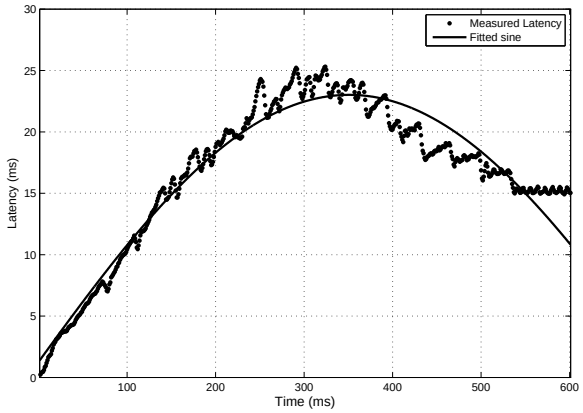
(f) Trail 6

Figure 4.13: Raw measurements of latency with fitted sine curve for trials with 50° rotational motion.

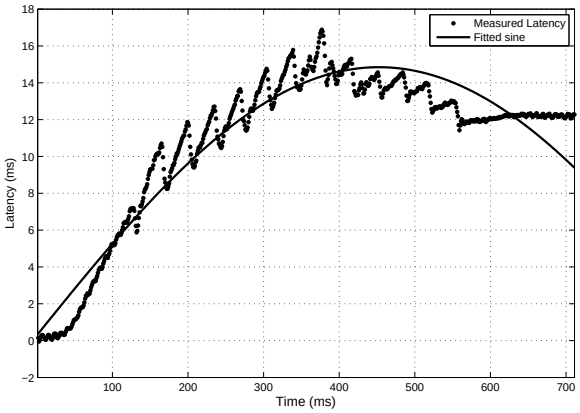


(a) Trail 7

(b) Trail 8



(c) Trail 9

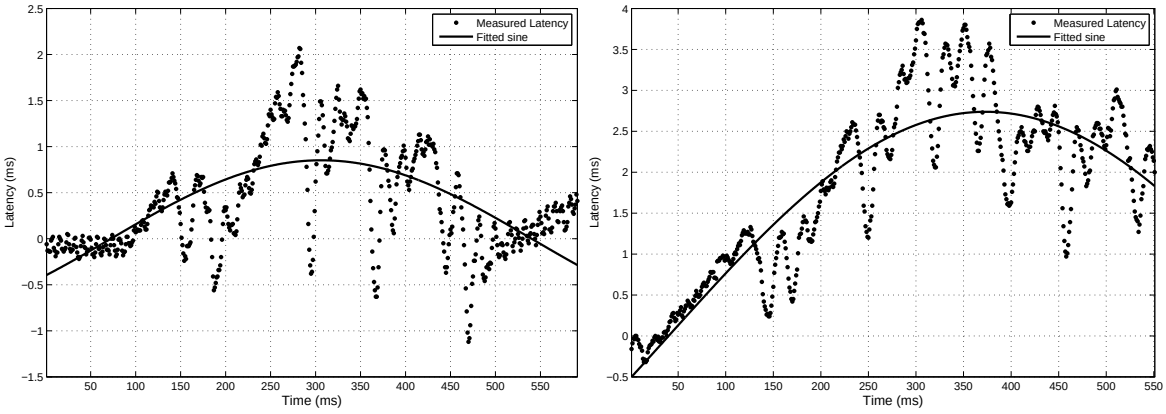


(d) Trail 10

Figure 4.14: Raw measurements of latency with fitted sine curve for trials with 50° rotational motion.

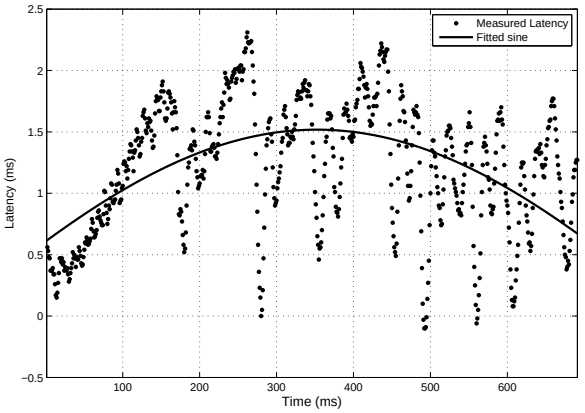
Latency (ms)	Frequency (Hz)
32.3	1.08
111.5	0.34
74.9	1.12
58.6	2.27
57.8	0.46
33.2	0.70
76.8	0.73

Table 4.2: Frequencies and magnitudes for apparent variations in latency, for seven trials with 10° rotational motion.

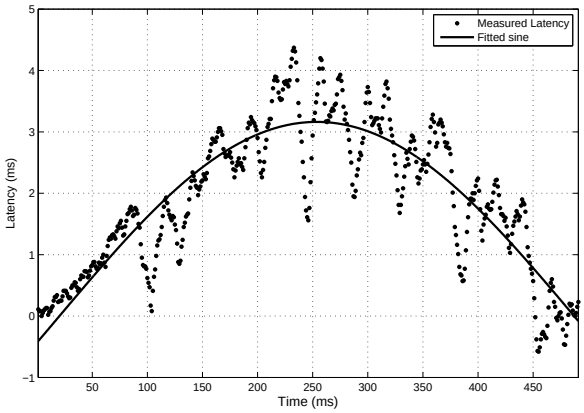


(a) Trail 1

(b) Trail 2

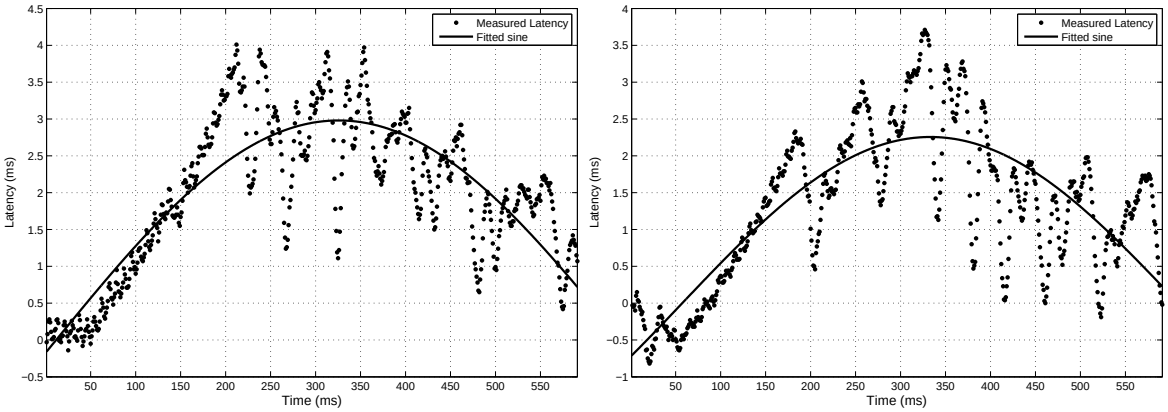


(c) Trail 3



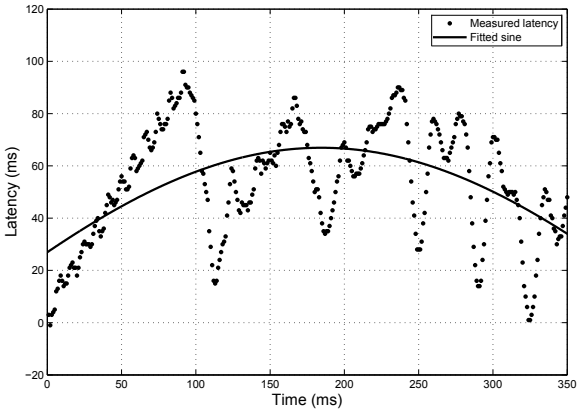
(d) Trail 4

Figure 4.15: Raw measurements of latency with fitted sine curve for trials with 10° rotational motion.



(a) Trail 5

(b) Trail 6



(c) Trail 1

Figure 4.16: Raw measurements of latency with fitted sine curve for trials with 10° rotational motion.

Chapter 5

Conclusion

In this thesis we have described a new method for measuring system latency. The main advantage of our method is that it measures latency continuously at 1 ms resolution. This allows for the observation of changes in latency over sub 1-second intervals of time. While many other works in this area have measured latency at an accuracy comparable to our method, the standard practice has been to calculate averages of repeated measurements. Figures 4.1, 4.9 and 4.10 show the types of information our method can reveal that cannot be seen in simple averages. We have found that differences in the clock frequencies of sensors and actuators cause a cyclical variation in latency; for the components we tested this was in the range of 15-50 Hz at magnitudes of 10-20 ms. We have also found that the error drift in sensor readings causes variations in apparent latency. For the orientation sensor we tested, which is popular in virtual reality and robotics research, the apparent variation in latency was in the range of 0.5-1.0 Hz at magnitudes of 20-100 ms. This magnitude of latency is known to be perceivable by humans, and this range of frequencies is known to be near the frequency that causes maximum sickness in humans [4]. Based on our results, we hypothesize that tracking system error may be at least partially responsible for the known phenomenon of "simulator sickness", where users of head mounted displays and virtual reality systems experience nausea or sick feelings while using these systems.

Adelstein et al. [1] and Di Luca et al. [9] have both previously noted that system latency is not a constant. They suspected that the type of motion, and especially its frequency, affected the latency. The hypothesized cause was filters used in the tracking system to smooth and predict the motion. Our work agrees with theirs, that different motions can cause different amounts of latency.

However, we believe it is the error in the tracking system that specifically causes the change in latency. Presumably if a tracking system erred similarly for multiple trials of the same motion, then a correlation would be found. This may partly explain their findings. For our system #2 tests, we did not pursue this idea, but in our limited trials we observed noticeably different sensor errors. A larger number of trials needs to be performed to more fully explore this possibility.

The methods of Miller and Bishop [11] and Di Luca et al. [9] may be modifiable to measure latency continuously. In particular, the method of Di Luca et al. [9] could presumably be operated at tens of KHz. In the future it would interesting to combine our approaches. This would allow for the evaluation of latency in systems that use sensors and actuators operating in the KHz range. In order to achieve this, it would be necessary to avoid using any high-pass filtering (as described in [9]), which removes variations of the type we are measuring, and to avoid using correlations for measurements, which only calculates averages.

Bibliography

[1] B. Adelstein, E. Johnston, and S. Ellis. Dynamic response of electromagnetic spatial displacement trackers. *Presence: Teleoperators and Virtual Environments*, 5(3):302–318, 1996.

[2] Y. Akatsuka and G. Bekey. Compensation for end to end delays in a vr system. In *Proc. of IEEE Virtual Reality Annual Int'l Symp*, pages 156–159, 2006.

[3] S. Bryson and S. Fisher. Defining, modeling, and measuring system lag in virtual environments. In *Proc. of SPIE - Int'l Society for Optical Engineering*, volume 1257, pages 98–109, 1990.

[4] J. Golding, D. Phil, A. Mueller, and M. Gresty. A motion sickness maximum around the 0.2 hz frequency range of horizontal translational oscillation. *Aviation, Space and Environmental Medicine*, 72(3):188–192, 2001.

[5] D. He, F. Liu, D. Pape, G. Dawe, and D. Sandin. Video-based measurement of system latency. In *Proc. of Int'l Immersive Projection Technology Workshop*, 2000.

[6] J. Liang, C. Shaw, and M. Green. On temporal-spatial realism in the virtual reality environment. In *Proc. of 4th Annual ACM Symp. on User Interface Software and Technology*, pages 19–25, 1991.

[7] V. Lippi, C. Avizzano, D. Mottet, and E. Ruffaldi. Effect of delay on dynamic targets tracking performance and behavior in virtual environment. In *Proc. of 19th IEEE Int'l Symp. on Robot and Human Interactive Communication*, pages 446–451, 2010.

[8] Y. Liu, A. Hoover, and I. Walker. A timing model for vision-based control of industrial robot manipulators. *IEEE Trans. on Robotics*, 20(5):891–898, 2004.

[9] M. Di Luca. New method to measure end-to-end delay of virtual reality. *Presence: Teleoperators and Virtual Environments*, 19(6):569–584, 2010.

[10] J M. Olano, Cohen, M. Mine, and G. Bishop. Combatting rendering latency. In *Proc. of 1995 Symp. on Interactive 3D Graphics*, pages 19–24, 1995.

[11] D. Miller and G. Bishop. Latency meter: A device to measure end-to-end latency of VE systems. In *Proc. of Stereoscopic Displays and Virtual Reality Systems*, 2002.

[12] M. Mine. Characterization of end-to-end delays in head-mounted display systems. *Technical Report TR93-001*, 1993.

[13] A. Morice, I. Siegler, and B. Bardy. Action-perception partterns in virtual ball bouncing: Combating system latency and tracking functional validity. *Journal of Neuroscience Methods*, 169:255–266, 2008.

[14] D. Roberts, T. Duckworth, C. Moore, R. Wolff, and J. O'Hare. Comparing the end to end latency of an immersive collaborative environment and a video conference. In *Proc. of 13th IEEE/ACM Int'l Symp. on Distributed Simulation and Real Time Applications*, pages 89–94, 2009.

[15] A. Steed. A simple method for estimating the latency of interactive, real-time graphics simulations. In *Proc. of ACM Symp. on Virtual Reality Software and Technology*, pages 123–129, 2008.

[16] C. Swindells, J. Dill, and K. Booth. System lag tests for augmented and virtual environments. In *Proc. of 13th Annual ACM Symp. on User Interface Software and Technology*, pages 161–170, 2000.

[17] R. Teather, A. Pavlovych, W. Stuerzlinger, and I. MacKenzie. Effects of tracking technology, latency, and spatial jitter on object movement. In *Proc. of IEEE Symp. on 3D User Interface*, pages 22–230, 2009.

[18] K. Waller, A. Hoover, and E. Muth. Methods for the evaluation of orientation sensors. In *Proc. of World Congress in Computer Science, Computer Engineering, and Applied Computing*, 2007.

[19] C. Ware and R. Balakrishnan. Reaching for objects in VR displays: Lag and frame rate. *ACM Trans. on Computer-Human Interaction*, 1(4):331–356, 1994.