ASSISTING HUMAN MOTION-TASKS WITH MINIMAL, REAL-TIME FEEDBACK

A Thesis

Submitted to the Faculty

in partial fulfillment of the requirements for the

degree of

Doctor of Philosophy

in

Computer Science

by

Paritosh A. Kavathekar

DARTMOUTH COLLEGE

Hanover, New Hampshire

June 2011

Dartmouth Computer Science Technical Report TR2011-695

Examining Committee:

(chair) Devin Balkcom

Andrew Campbell

Chris Bailey-Kellogg

Katsu Yamane

Brian W. Pogue, Ph.D.
Dean of Graduate Studies

Abstract

Teaching physical motions such as riding, exercising, swimming, etc. to human beings is hard. Coaches face difficulties in communicating their feedback verbally and cannot correct the student mid-action; teaching videos are two dimensional and suffer from perspective distortion. Systems that track a user and provide him real-time feedback have many potential applications: as an aid to the visually challenged, improving rehabilitation, improving exercise routines such as weight training or yoga, teaching new motion tasks, synchronizing motions of multiple actors, etc.

It is not easy to deliver real-time feedback in a way that is easy to interpret, yet unobtrusive enough to not distract the user from the motion task. I have developed motion feedback systems that provide real-time feedback to achieve or improve human motion tasks. These systems track the user's actions with simple sensors, and use tiny vibration motors as feedback devices. Vibration motors provide feedback that is both intuitive and minimally intrusive. My systems' designs are simple, flexible, and extensible to large-scale, full-body motion tasks.

The systems that I developed as part of this thesis address two classes of motion tasks: configuration tasks and trajectory tasks. Configuration tasks guide the user to a target configuration. My systems for configuration tasks use a motion-capture system to track the user. Configuration-task systems restrict the user's motions to a set of motion primitives, and guide the user to the target configuration by executing a sequence of motion-primitives. Trajectory tasks assume that the user understands the motion task. The systems for trajectory tasks provide corrective feedback that assists the user in improving their performance. This thesis presents the design, implementation, and results of user experiments with the prototype systems I have developed.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Teaching physical motions such as riding, exercising, swimming to human beings is hard. Coaches face difficulties in communicating their feedback verbally and cannot correct the student mid-action; teaching videos are two dimensional and suffer from perspective distortion. I have developed motion feedback systems that provide real-time feedback to achieve or improve human motion tasks. This thesis presents the design, implementation, and results of the systems that I have developed.

Consider a person learning to ride a bicycle. In a typical beginner's mistake, the rider's weight shifts slowly to one side. This weight shift causes the bicycle to lean, eventually causing the rider to fall down. As another example, consider the exercise of lifting weights. In many cases, this exercise leads to injuries because of poor technique, which repeatedly stresses the lifter's body.

In the examples above the users are *inside* the system; they cannot see or identify their mistakes. The bicycle rider's weight shifts slowly, making it difficult for him to notice the shift. Over time, these errors accumulate leading to the fall. Similarly, the injuries due to bad form while lifting weights add up over time.

External aids that improve performance are already used for a variety of applications: musicians use metronomes to maintain a constant tempo, distance runners use other runners as pacemakers to maintain speed, etc. My experiments with prototype systems indicate that real-time, corrective feedback enables users to attain motion task goals and improve motion accuracy.

Systems that track a user and provide him real-time feedback have many potential applications. For example, such systems can be used as an aid to the visually impaired, helping them in common everyday tasks. Rehabilitation after accident or illness often requires a certain sequence of exercises that have to be performed adequately. Systems that track and improve the quality of such actions can benefit patients and medical practitioners. Many forms of physical activity, such as lifting weights, require maintaining correct form. Feedback that aids the user in maintaining good form could improve the effectiveness of such exercises without the need for a human trainer. Other potential applications include training a large group of people to match an exemplar motion—synchronizing dancers in a troupe, or teaching programs that improve swimming by matching a user's strokes to those of an expert's.

It is not easy to deliver real-time feedback in a way that is easy to interpret, yet unobtrusive enough to not distract the user from the motion task. I have built a set of simple systems that can be used for training and sensory augmentation. These systems track the user's actions with simple sensors, and use tiny vibration motors as feedback devices that guide the user to the target configuration. Vibration motors provide feedback that is both intuitive and minimally intrusive. My systems' designs are simple, flexible, and extensible to large-scale, full-body motion tasks.

1.1 Contributions and goals of this thesis

The primary aim of this thesis is to develop real-time feedback systems for a broad range of motion tasks, and study their feasibility and limitations. In addition to developing these systems, this thesis aims to study the applicability of different devices in tracking users and providing feedback. Chapter 3 provides details about the devices that I used, and their advantages and limitations. The final aim for the thesis was to explore different algorithms for providing feedback. Different motion tasks have different feedback needs, and there are many strategies to fulfil a given motion task. I developed a variety of controller algorithms to fulfil a range of motion tasks and in some cases developed multiple algorithms for the same task.

To fulfil these goals of, I developed real-time feedback systems for a broad range of motion tasks, measured their performance and analyzed their limitations. My systems track users with body-mounted sensors and use small vibration motors as feedback devices. A controller program running on a computer calculates the user's state from the sensor data and provides motion feedback using the vibration motors. Using this setup, I developed systems that provide feedback for the following motion tasks:

- **Mobile manipulation system**: I developed a system that enables a blindfolded user to navigate an obstacle-free area, and manipulate objects placed in the environment. This system uses a motion-capture system to track the user and comprises two subsystems: the navigation subsystem and the manipulation subsystem. The navigation system guides the user to the target position and orientation, and the manipulation subsystem guides the user's hand to the target position.

  The mobile manipulation system restricts the user's actions to a small set of motion primitives. Motion primitives are natural motions that need minimal effort to learn. Further, this system prompts the user to perform only one motion from the set of motion primitives at any time. Chapter 4 describes this *restricted feedback* in detail. The mobile manipulation system prompts the user through a sequence of motion primitives to achieve the task.

  I tested this system on a blindfolded user. During the experiment, the (blindfolded) user was guided to a location where the manipulation subsystem prompted him to pick an object placed close by. The user was then guided to a target location, and instructed to place the object at a target position. The mobile manipulation system is discussed in detail in chapter 4.

- **System for posing a four degree-of-freedom arm**: I developed a system that guides the user to attain a pose for a four degree-of-freedom arm: three degrees of freedom in the shoulder joint, and the bending of the elbow. Unlike the manipulation system where the target configuration corresponds to the position of the hand placing the object, a pose defines the configuration of the entire arm.

  The arm-posing systems uses a motion-capture system to track the user(s). Like the mobile manipulation system, the arm-posing system uses restricted feedback and a sequence of motion primitives to guide the user's arm to the target pose.

  I tested the arm-posing system under two conditions. In the first experiment, the user's arm was guided to a pre-specified pose. In the second experiment, the motion capture system tracked two users: a leader and a follower. The leader assumed poses unknown to the follower, and the posing system guided the blindfolded follower to attain the pose assumed by the leader. Chapter 5 presents the design and implementation of the arm-posing system.

- **A shirt-folding system**: I developed a system that allows a blindfolded user to fold a T-shirt. This system allows a user to manipulate a T-shirt with both hands and fold it using a sequence of predefined motions.

  The motion primitives for the shirt-folding system correspond to natural arm motions like moving the arm up/down instead of rotation around a joint axis. The shirt-folding system extends the manipulation system to control two arms simultaneously. This system models the shirt folding process as a sequence of manipulation tasks, where the target configuration for one task depends on the target configuration for the previous task. The dimensions and position of the T-shirt are the input for this system, and the controller calculates the target configurations for each step at runtime.

  The shirt-folding system demonstrates that complex tasks can be accomplished using simple tracking and feedback. This system also validates the design decision of leveraging human being's natural abilities for manipulating soft objects such as cloth. Chapter 6 presents the details of the shirt folding system.

- **A posture shirt**: Many motion tasks require users to maintain a proper posture, *e.g.* yoga poses need to be *held* for a certain period. Similarly, people working at a desk find it difficult to maintain the ideal posture without corrective feedback. I developed a system that assists users to maintain a torso posture while sitting down.

  This system, developed as a wearable shirt, uses a three-axis accelerometer to detect the user's torso configuration, and provides feedback when the posture changes beyond a threshold. This system prevents the user from bending too far forward (or backwards) and leaning too much sideways. The posture shirt provides a restraining feedback, and the design allows small movements beyond the threshold (for picking up objects, for example).

  I tested this system on volunteers who were asked to wear this shirt. My system improved the users' posture by significantly reducing the time the user's posture was outside the specified threshold. Chapter 7 presents the design, implementation, system results, and user experience results for this system.

- **Arm-motion-controller system**: Many common motions consist of multiple segments that are iterated cyclically. I have developed a system that provides corrective feedback for such multiple segment, cyclic arm motions. My system provides corrective feedback for three aspects of arm motions: accuracy of the motion, timing between segments, and form of the rest of the body while performing the motion.

  This system uses accelerometers to track the user's arm. I tested this system on volunteers against two competing forms of feedback: verbal explanation of the motion, and showing them a video of the motion (before asking them to perform the motion). The results show that my system performed significantly better on all three aspects of the motion compared to systems that provided verbal instructions or visual instructions. I will present the details of this system in chapter 8.

- **A motion-synchronization system**: For many applications human beings have to synchronize their actions. Examples of such activities include rowing a boat, dancing in a troupe, etc. I developed a system that synchronizes simple torso motions between two users.

  My system uses a leader-follower paradigm, where one user acts as a leader and the second user, the follower, tries to match the leader's motion. The motion synchronization system used

accelerometers to track the torso configuration of both the users. The follower was provided a limited form of velocity feedback to match his configuration with that of the leader.

I chose a simple torso motion to test my system on volunteers, and compared my system against two other modes of feedback: open-loop instructions where I described the motion to the users but they could not see each other, and visual feedback where the follower could see the leader. For the simple torso motion of the experiment, the visual feedback acted as the *perfect* feedback case. The results show that my system performed well when compared with the other two systems. The design, implementation and results for the motion synchronization system are presented in chapter 9.

1.2 Spectrum of motion feedback systems

Many existing systems provide feedback that assist and teach humans. These systems are extremely diverse with respect to their applications, philosophies, and methods of providing feedback. One attribute of such systems that helps put my systems in perspective is the detail of feedback that such feedback systems provide.

The level of feedback for most motion feedback systems falls in two broad categories. On one end of the feedback spectrum are systems that provide very detailed feedback for a specialized task, *e.g.*, a haptic glove developed by Jack *et al.* [38] for rehabilitation. On the other end of the feedback spectrum are systems that provide extremely limited feedback, *e.g.*, NavBelts [72] that assist blind people with indoor navigation. Detailed feedback systems are used for tasks that need a high degree of precision. These systems are almost always used for a specialized task, and tend to be very intrusive. Low-level feedback system, on the other hand, are mostly used for tasks that do not need significant accuracy. These systems tend to be significantly less intrusive.

Many tasks need to be performed at an intermediate level of accuracy. Examples of such tasks include manipulating objects in an environment, calisthenic exercises, dancing, etc. Extremely limited feedback is insufficient for such tasks, but feedback from specialized, high degree of feedback systems is difficult and too intrusive. The systems that I have developed as part of this thesis are primarily aimed at tasks that require an intermediate level of feedback. An additional feature (discussed in section 1.4) of my systems is extensibility: the ability to develop systems for many-degree-of-freedom motion tasks.

1.3 Types of motion feedback

In this thesis, I have developed systems for two broad categories of motion tasks: configuration tasks and trajectory tasks. *Configuration tasks* aim to guide the user to a target configuration. For configuration tasks, the path taken from the initial configuration to the target configuration is not important; the motion feedback system can choose any path to guide the user to the target configuration. Examples of configuration tasks include moving and manipulating objects and indoor navigation. *Trajectory tasks* guide the user to follow a particular trajectory. For these tasks, the goal is defined as following a particular trajectory, instead of achieving a specific configuration. The two categories are not exhaustive. In chapter 9, I will describe details of a system that tries to control both the trajectory and the speed of the motion.

The complexity of a motion task is different from the complexity of the underlying motion. For example, walking is a complex motion that involves moving multiple joints and balancing the body's weight. However, a motion task defined as prompting a person to walk in a straight line

requires a simple motion controller. The complexity of the motion task affects the level of feedback needed to accomplish it.

Target configurations are the input to configuration-task system. My systems for configuration tasks assume that the user is constrained to a small set of motion primitives. Configuration-task systems guide the user to one or more target configurations using a sequence of motion primitives. Chapters 4, 5, and 6 provide details of motion feedback systems for configuration tasks.

For trajectory tasks, the underlying motion may be extremely complex. Providing comprehensive feedback for such motions (assuming no knowledge from the user) is challenging. The hardware and software required for such detailed feedback is difficult to build. Further, even if such a system could be built, the users may not be able to comprehend and react to the feedback in real time. Therefore, for trajectory systems, I have made a simplifying assumption: the user *understands* the underlying task, but is unable to perform it precisely. Systems for trajectory tasks provide *corrective feedback*. These systems provide feedback that assists the user in performing the task more accurately. In chapters 7, 8, and 9, I will present the details of systems that I developed for trajectory tasks.

1.4 System design and overview

From a system-design perspective my systems are close to closed-loop robotics control, where the human being replaces the robotic actuator. For configuration tasks, my systems define a set of motion primitives for the controller. The choice of motion primitives depends on the task. For trajectory tasks, my systems takes the target trajectory as input and guide the user towards it. My systems use low-information feedback that can be extended to the entire body, and requires very little learning on the user's part.

There are some fundamental differences between robots and humans from a controller's perspective. Humans have binocular vision, touch sensitive skin, ten fingers, compliant force control, and natural inverse kinematics. Conversely, due to lack of a "standard interface" human response to feedback is slow, error-prone, and unpredictable.

Our systems leverage human capacities for sensing and manipulating the environment. For example, the shirt-folding system (chapter 6) uses human abilities to naturally perform inverse kinematics and grasp cloth. These two tasks are extremely difficult for robots but relatively simple for human beings. Adding this human element to the control loop simplifies the design and implementation of my systems. My systems have tried to address some human limitations. These system provide feedback that is detailed enough to enable users to accomplish their tasks, yet intuitive and simple enough for users to comply with it.

An important characteristic of our systems is minimalism. Both while tracking and providing feedback, my systems observe and convey as little information as possible. While tracking the user, these systems only track a small number of key points (or joints) on the user's body. Similarly, the feedback systems conveys as little control instructions to the user as possible. Minimalism lets these systems scale to the entire body, because larger systems are not intractable to design and build. Also, low-information feedback is easy to interpret and less intrusive than modes of feedback that require constant user attention.

Figure 1.1 shows a high-level view of my systems. My systems consist of three major components apart from the user: the tracking system, the feedback system, and the controller. The tracking devices provide the controller with the user's current state. Based on the user's state the controller calculates the appropriate feedback and conveys it to the feedback device. The user changes his state based on this feedback, and the system iterates till the user completes the motion task.

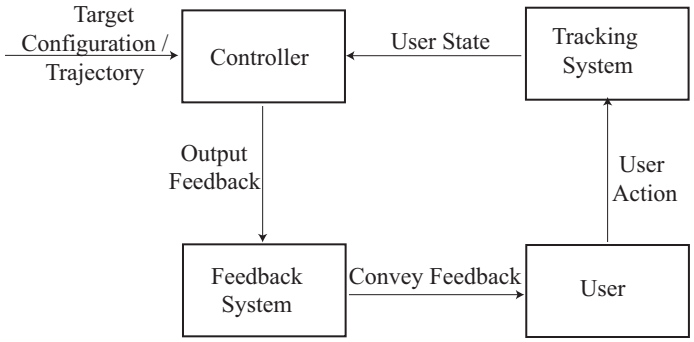


Figure 1.1: Overview of our system. The tracking system tracks the user and sends the current state to the controller. The controller calculates the required feedback and sends it to the user, who changes state according to the feedback

The tracking system comprises devices that track the user and convey his present state to the controller. As mentioned before, my systems follow the principle of minimalism while tracking the users: they track only the relevant information about the user.

My systems provide vibrotactile feedback using Lilypad vibration motors [13] that are mounted on the user. Based on the controller's instructions, some of these motors are switched on or off. The user performs his actions depending on the motors that are buzzing. To control the vibration motors the controller sends the instructions to a communication device. The communication devices, in turn, control the feedback motors. Chapter 3 discusses the details of the tracking, feedback and communication devices that I used for developing my systems.

The controller performs a variety of functions to achieve its objective. First it interacts with the tracking devices and obtains their sensor data. The controller filters this usually noisy data, and calculates the user's state. Based on the user's current state and the target state the controller calculates the necessary feedback. It then conveys this feedback to the feedback system via the communication devices. My controllers are written in C++ and run on a Windows workstation.

1.5 Limitations and future work

This thesis is aimed at developing and analyzing prototype systems for a broad range of motions. The primary goal is to demonstrate the feasibility and potential of real-time motion feedback systems. There are many other aspects of developing and analyzing such systems that were not the goals for this thesis. However, these are important open problems that need to be studied in the future. This section presents some limitations of this thesis and highlights some broad directions for future work.

- **Explore and analyze all tracking and feedback systems**: There are many different sensors available to track users such as marker-based motion capture, markerless motion-capture, accelerometers, compasses, inertia measurement units. Similarly there are many different methods of providing feedback such as visually, aurally, virtual-reality based, offline, vibrotactile feedback. The best method for tracking and feedback depends on the task. Testing and analyzing all methods of tracking and feedback was not a goal for this thesis. However, testing and analyzing a variety of tracking sensors and feedback devices is important for developing new applications, and should be an important area of study going forward.

- **State-of-the-art systems**: My systems accurately track users and provide sufficient feed-

back to achieve their objectives. However, other methods to interpret sensor data might improve tracking. Similarly, alternate feedback strategies might improve accuracy of human compliance with the feedback.

The systems that I have developed have direct practical applications. My goal for this thesis was to develop prototype applications for a broad range of motion tasks rather than state-of-the art, commercial-grade systems for a narrow range of motions. There are many practical hurdles such as more robust tracking, reducing the setup time, and making the systems more usable, that need be overcome before my systems can be deployed on a large scale.

- **Tools for comparing motions**: Comparing motions and measuring how close two motion trajectories are is a fundamental requirement for some motion tasks. In this thesis, I did not develop general purpose mathematical tools for comparing two trajectories, *e.g.*, a metric for comparing two motion trajectories expressed in joint space. Systems that intend to evaluate the performance of individuals on specific tasks would benefit from a general framework of comparing motion trajectories.

- **Teach people motions**: My systems are aimed at achieving or improving a given motion task. Teaching a motion implies that the improvements caused by the feedback persist even when the feedback is removed. Measuring such improvement needs separate experiments that run over a longer duration. I did not perform those experiments for this thesis, but these experiments are essential to measure the effectiveness of systems that are designed to speed up the learning of new motion tasks.

- **Other applications and controller algorithms**: I have developed motion feedback systems for a broad range of applications that use a variety of controller algorithms. However, this set of applications and feedback algorithms is not exhaustive. Other applications might benefit from real-time feedback. Similarly, other controller strategies might be more suitable in some cases, maybe even for motion tasks that I have addressed as part of this thesis. Developing new applications and new controller strategy is a promising area for future research.

All the goals mentioned above are worthy aims to pursue. They address issues that are no less important than the ones I have addressed in this thesis. However, this thesis aims to determine the broad applicability of real-time motion-feedback systems. The goals stated above assume greater significance once the feasibility of such systems is established.

Chapter 2

Related Work

This chapter summarizes work that is related to several problems that I directly or indirectly address in this thesis. I have developed systems that provide feedback for human motion tasks. These systems have three basic components: sensors that track the user, feedback device that provide feedback, and controller programs that determine and communicate the appropriate feedback. I have used two sensors to track the user: an optical motion capture system and three axis accelerometers. All my systems provide vibrotactile feedback using Lilypad vibration motors [13], though the details of the feedback differ between systems. Chapter 3 presents details of the hardware devices that my systems use. My systems provide feedback for two broad categories of motion tasks: configuration tasks and trajectory tasks (described in chapter 1).

Many researchers have developed systems that track a user and provide task-specific feedback. These systems differ from each other in their applications, sensors used for tracking the user, and their feedback devices. I will classify motion feedback systems according to their feedback type, and discuss the major classes of feedback. Systems that provide vibrotactile feedback are the closest to the ones I have developed. These systems are discussed in section 2.1. Virtual-reality based systems are surveyed in section 2.2, followed by haptic systems in section 2.3. Finally, in section 2.4 I will survey systems that provide offline feedback.

Motion capture systems are the most common method for tracking users because of their convenience. However, many systems, including some that I developed, use alternate sensors to track users. These sensors offer many advantages over motion capture systems, such as cost and portability. Section 2.5 describes systems that track human motions with different sensors. Some of these systems do not provide feedback, but they provide opportunities for future applications.

Training physical motions is an important application for motion feedback systems. To measure their effectiveness, motion training systems need algorithms and metrics to compare two motion trajectories. Although not a part of this thesis, motion training systems and consequently comparing motion trajectories are important areas for future research. Section 2.6 covers approaches for comparing motion trajectories.

My systems for configuration tasks employ restricted feedback; they restrict the user's motion to a small set of motion primitives and allow the user to perform only one of those motion primitives at any time. These restrictions prevent the user from following a general trajectory in configuration space. For systems that use restricted feedback, algorithms that approximate a given trajectory closely are important. Approximating general trajectories with restricted feedback is closely related to the problem of polygonal approximation. Section 2.7 presents some previous work in the area of polygonal approximation.

2.1 Vibrotactile-feedback-based systems

My systems provide vibrotactile feedback. Vibrotactile feedback typically uses vibration motors that prompt the user to perform the target motion. This section describes motion feedback systems that use vibrotactile feedback.

The TIKL system [46] is perhaps the closest to my systems. TIKL augments visual feedback with feedback from vibrotactile motors. This system captures a user's state with an optical motion-capture system. Users that tested this system were first shown short actions requiring between one to five degree of freedom that were projected on a computer screen. They were then asked to repeat the actions, and in addition to seeing their current state (captured with a motion capture device) on the computer screen, they were also given tactile feedback to correct the motion. Augmenting visual feedback with tactile feedback improved the learning of certain motions.

Although TIKL uses optical motion capture and provides vibrotactile feedback, it differs from my systems in many ways. TIKL uses vibrotactile feedback to augment visual feedback, whereas my system use only vibrotactile feedback. My systems for configuration tasks (systems that use the motion capture system) are goal-oriented. Hence, the path followed to the target configuration is not important. On the other hand, TIKL only focuses on following exact trajectories. Finally, TIKL focuses on a limited range of arm motions, whereas this thesis presents systems for a broad range of applications such as mobile manipulation, arm posing, posture shirt, and motion synchronization.

Navbelts are a popular form of electronic travel aid for the blind. These devices were originally developed in the 1990s ([11], [72]) and are still popular with users. A Navbelt is a belt with many vibration motors. These devices provide navigation feedback by selecting the motors that vibrate. Newer Navbelt systems combine these vibration motors with auditory feedback and use sensors to detect obstacles.

In [76], Spelmezan *et al.* studied the feasibility of providing vibrotactile feedback for whole-body motions. Their results show that users can identify and select between a variety of expected motions using vibrotactile feedback. In their experiments, users reacted better and faster to vibrotactile feedback than to auditory feedback.

Zheng and Morrell [90] designed a posture chair. Their system uses seven force sensors embedded in the chair to detect the user's posture and six vibrotactile motors to provide posture feedback. The posture chair can identify ten postures for the user's back and legs (one sitting straight and nine anomalous postures) and provides corrective feedback.

Vibrotactile feedback is rapidly gaining in popularity in the domains of sports, navigation, rehabilitation, gaming, and motor learning. Alakhone *et al.* [1] provide a recent survey of applications using vibrotactile feedback.

Although the systems presented in this section track a user's action and provide real-time feedback, they are focused on training for a particular motion. Most of these systems do not leverage human capabilities. My systems refrain from elaborate tracking or feedback mechanisms because it is hard to use such systems in real-world settings. Leveraging human capabilities simplifies design and improves usability.

2.2 Virtual-reality based training systems

Virtual-reality (VR) systems create a computer-graphics-generated *virtual environment*. Using a head-mounted display, a user can visualize and interact with users and characters in the virtual environment. VR-based systems are often used as simulators or trainers for environments that are dangerous, or difficult to reproduce in real world.

One class of VR applications are used to train users in situational awareness. Examples of such systems include [70], used to train law enforcement personnel for hostage crises. A similar system, described in [23], trains users to evacuate a battleship in case of emergencies. By repeatedly exposing the user to a particular environment, these systems familiarize the user with different potential scenarios, and train him to take the correct course of action.

Another class of VR trains users with intelligent *agents*. Agents are virtual characters in the virtual environment that interact with the user and train them. Rickel and Johnson ([39],[67]) created an interactive agent STEVE (Soar Training Expert for Virtual Environments) to assist users with machine operation training. STEVE responds to verbal queries, tracks users' actions and demonstrates correct procedures when requested. These systems are highly specific and their mode of feedback is verbal instruction.

Holden *et al.* ([48], [34]) describe a virtual-environment-based tracking system to augment conventional rehabilitation therapy. In their system, the patient is asked to imitate certain motions that are streamed to her via video. A motion-capture system captures the patient's motion, and feeds them in real time to a nurse who watches these actions superimposed with the ideal motions in a virtual environment. By studying the difference between the two motions, the nurse can provide effective feedback to the patient. Piron *et al.* ([62], [63]) have applied a similar technique for upper-arm motion rehabilitation with encouraging results. Interestingly, in these systems, the nurse and the patients do not need to be at the same facility. The patient performs the exercises at home, while the nurse may be at work monitoring over the internet. Like my systems, these systems track a user and provide feedback in real time. However, these systems require humans to control the input stream and generate feedback.

The *Just follow me* system [86] uses *ghost metaphors* with virtual-reality to teach users dance moves. A ghost metaphor is a virtual-reality image of a trainer (constructed from motion-capture data) that is displayed in front of the trainee, who is expected to imitate it. Hachimura *et al.* [30] developed a similar system using mixed-reality technology. This system captures the trainee using motion-capture in real time. The trainee's reconstructed skeleton is then superimposed on the expert's skeleton using computer graphics and VR technology.

2.3 Haptic feedback systems

Haptic devices provide force feedback to guide or correct human motions. Haptic feedback is closer to our feedback devices than VR-based feedback. However, most haptic feedback systems tend to be task-specific, because the type of reaction forces depend on the particular application.

Motion training is an important part of rehabilitation following a stroke or neural injury. MIT-Manus [33] (and later generations of the same system) are examples of haptic devices that learns a particular action in a learning phase and guide the patients through the same motion. Manus has a five degrees of freedom SCARA arm, and the patients need to physically connect to the robot to follow the required trajectory. Volpe *et al.* [82] show encouraging results using the Manus robot in rehabilitation of patients following a stroke.

Exoskeletons are external devices worn by users that are popular for rehabilitation applications. These systems measure the user's current state and provide force feedback as the user performs a motion. The Bionic research lab at University of California at Santa Cruz has developed a wide range of exoskeletons for the human arm. Their most detailed exoskeleton for the upper body allows the user to control a seven-degree-of-freedom human arm [60] (three degrees of freedom in the shoulder, one at the elbow and three in the wrist). Other assisting systems such as [19] target the lower limbs with special emphasis on gait correction. Exoskeletons have many potential

applications such as haptic devices, human amplifiers, physiotherapy, assisting users with tasks.

Some attempts have been made to leverage the power of virtual-reality systems for haptic feedback. Jack *et al.* [38] describe a novel rehabilitation approach where patients wear haptic gloves to perform predesigned tasks in a virtual world. Yokohoji *et al.* [88] designed a system (WYSIWYF) to provide haptic feedback in a virtual environment. They track the user's action with optical motion capture, and estimate the hand position in the virtual world. If the user touches an object in the simulated environment, their system provides the appropriate touch feedback.

2.4 Motion tracking systems with offline feedback

Motion capture systems provide a convenient and accurate method to track subjects as they perform a specific task. Many systems capture and compare both experts and novices as they perform the same action. Such comparisons can help users identify differences between exemplar and novice motions.

Alexander *et al.* [2] used simple, low-resolution video-based motion-capture to analyze the efficacy of exercise in the elderly. Their system computes the contours of the subject's back and shoulder as they walk on a treadmill. This data is analyzed offline to provide corrective feedback.

Mora *et al.* [51] have used motion-capture data for correcting body posture while playing the piano. Their system uses a motion-capture system to record the postures of a master and a beginner artist while playing the piano. In the analysis phase, they superimpose 2D videos of the captured skeletons. The user can view the superimposed videos from multiple viewpoints and analyze the differences in posture.

Motion-capture systems have been extensively used to model and analyze human joint-motions, especially in the domain of sports [52]. The product website for Vicon systems [80] provides a comprehensive (yet not exhaustive) list of applications developed using Vicon's motion capture systems. Most systems described there focus on a narrowly-defined task, and provide offline feedback.

2.5 Alternative methods to track user and applications

Conventional motion-capture systems have limitations. These systems are expensive, which limits their large-scale use, and the user is confined to a small capture volume. Typically, these systems use high-speed, high-resolution cameras that provide extremely accurate measurements (around 1mm). Many applications do not need such precise measurements, and can perform adequately with less accurate, less expensive, and more flexible systems. Consequently, many researchers have developed systems that use alternate sensors to track human motions. These new systems are cheaper and are targeted towards particular applications. In this section, I will briefly describe the design and applications of some of these systems.

A significant proportion of these alternate tracking systems use accelerometers to track human body segments. An accelerometer provides the magnitude and direction of the acceleration the sensor experiences. However, integrating the acceleration data to get position is error prone because of drift errors. A range of solutions exist to counter these errors. I will specify the particular error-correction method for systems that use accelerometers.

Lee and Ha [44] describe a system to track humans using three-axis accelerometers. They use a *sensor-fusion* technique to correct drift errors. This method places more than one sensor per segment, and uses the average sensor value to calculate and compensate for the drift errors. Bachmann *et al.* [5] designed the MARG sensors to correct the drift problem in tracking human motions. MARG sensors have three micromachined rate-sensors, accelerometers, and magnetometers each (a

total of nine sensors) for accurate measurement of human motions. Yun *et al.* [89] used accelerometers and a compass to accurately track actors' position as they walked. They filter drift errors between steps by careful calibration and mathematical techniques.

Vlasic *et al.* [81] designed a portable motion-capture sensor system that can track people anywhere. Their system uses a combination of accelerometers, gyroscopes, and an acoustic subsystem to accurately follow the motion of different body segments. The acoustic sensors measure the distance between limbs and counter the drift from accelerometers.

Prakash [65] is an alternate optical motion-capture system. This system uses photosensitive markers that can be embedded in clothing. The markers are individually identifiable (because they are tagged with RFID); therefore, the system can be scaled to any extent. This system is significantly cheaper and more flexible than conventional motion-capture systems, because it does not use high speed cameras.

The NorthStar system [55] from Evolution Robotics provides simple position and orientation tracking. This system projects IR rays on reflectors that are placed on the ceiling. An IR detector on the tracked object measures these reflections and uses triangulation to calculate the tracked object's position.

Animating virtual characters is another application that uses alternate tracking systems. Perhaps the most well-known examples of such systems are the Wii gaming consoles [83] from Nintendo Inc. and Microsoft's Kinect [42] gaming console. Wii uses data from a three-axis accelerometer embedded in a remote that is held by the user to animate characters in a video game. Kinect uses two cameras to track the user in the capture volume.

Slyper and Hodgins [74] developed a systems that uses five accelerometers attached to a user to animate an avatar. Their system precalculates the acceleration data for a motion capture database. At run time, the acceleration data from the user is compared with the acceleration data stored in the motion capture database to identify the user's action, and animate the avatar.

MOCA (Farella *et al.* [25]) is a gesture recognition system for VR avatars that uses accelerometers. This system relies on the fact that very accurate motion data is not needed to identify the correct gesture from a given set. MOCA identifies the intended gesture by comparing the user's acceleration data with a database of gestures with known accelerations.

In a similar application called *Footsee*, Yin and Pai [87] recognize the patterns of a user's footsteps to animate an avatar in VR. During the training phase, their system creates a motion-capture database for a range of motion and the corresponding ground pressure distribution on a pressure pad. At run time, the user's footsteps pattern is measured and compared with the database to find the correct action. The matched motion is used to animate an avatar.

Johnson *et al.* [40] built a *sympathetic interface* to let users control the actions of an animated character. Their system captured the user's action using a plush toy embedded with wireless sensors such as accelerometers, magnetometers, and gyroscopes. Users move the toy to indicate their intention. Depending on the user's feedback and the context, the system animates the virtual character.

In a novel medical application, Lee *et al.* [45] estimated the gait parameters such as stance, swing, single support, and double support time of the gait cycle using accelerometers attached to the patient's ankles.

Hesch *et al.* [32] used a pedometer, a walking cane, a three-degree-of-freedom gyroscope and a 2D laser scanner to guide a blind user in a known environment. Their system mounts the gyroscope and laser scanner onto the cane and uses a two-stage, Extended-Kalmann-Filter-based algorithm to estimate the user's current position. Zheng and Morrell [90] used force sensitive resistors to track a user sitting in a chair.

2.6 Comparing trajectories

Motion-training systems often compare trajectories, *e.g.*, to show improvements after using the system. However, comparing motion trajectories is challenging because of difficulties such as the lack of standard metrics for comparing trajectories and finding the optimal mapping between trajectory points to compare. As a result, there are no standard methods to compare trajectories. Developing methods for comparing trajectories is an open research problem that is especially relevant for motion training systems. In this section, I will describe the problem of comparing motion trajectories as it appears in several different contexts, and the approaches taken to solve this problem.

Trajectory matching in two dimensions

Comparing and/or identifying motion trajectories for planar curves is a well-studied problem. In this setting, the motion trajectory is represented as a parametric set of points $(x(t), y(t))$ in the XY plane, where t represents time. This problem is closely related to the problem of matching curves in two dimensions, where time is replaced by other parameters such as the curve length.

One popular method of comparing trajectories uses motion descriptors. A descriptor is a compact representation that uses a small number of parameters to represent a trajectory. The Fourier Descriptor that represents a trajectory using its 2D Fourier coefficients is one such example.

Researchers have developed several descriptors for comparing curves. Harding *et al.* [31] and Bashir *et al.* [7] used variants of Fourier Descriptor to store and classify motion trajectories. The curvature space similarity descriptor (CSS) by Mokhtarian *et al.* ([49], [50]) and wavelet descriptor [15] are examples of hierarchical descriptors. Hierarchical descriptors represent an image from a coarse to fine resolution. CSS converts trajectory data to curvature data before calculating the descriptor.

Other descriptors such as moment invariants [35] and B-spline decomposition [17] compare the global properties of the motion such as the central moments of the curve, or the coefficients of the B-spline decomposition of the curve.

In the field of computer vision, model-based approaches are used for tracking and segmentation of motion trajectories, *e.g.*, Fleet *et al.* [26]. These applications first learn a "basis" set of human motion with learning techniques such as PCA. Then, they used the learned basis to classify and segment motion.

Segmenting motion capture data

High-level motions such as walking or moving hands consist of shorter, more basic segments. Separating such segments requires identifying the *transition points*, the points that separate two motion segments.

Fod *et al.* [27] designed a system to train a robot manipulator to move like humans. They tracked human motions and segmented it into basic segments that could act as building blocks for more complex motions. Their method used simple zero-point crossings of joint velocities to segment motion trajectories.

Kovar *et al.* [43] and Brand and Hertzman [12] developed motion segmenting algorithms for computer graphics applications. Their applications create new motions from existing motions segments by identifying compatible segments to transition from one type of motion segment to another.

Barbič *et al.* [6] address the problem of segmenting higher-level motions. Their data consists of a sequence of high-level motion primitives such as walking, running, jumping, etc. The segmentation

algorithm relies on the fact that the high-dimensional motion data lies on a lower-dimensional subspace. Barbič *et al.* describe three approaches for segmenting the data: PCA, probabilistic PCA and Guassian Mixture Models.

Comparing motion trajectories for training

In an application partially geared towards robot programming by demonstration, Wu, Li and Zheng [84] describe a hierarchical motion-trajectory descriptor for three-dimensional curves. Their descriptor describes a three dimensional trajectory with its curvature and torsion at each point. They used Dynamic Time Warping (DTW) to find an optimal matching of trajectory configurations in the descriptor space. Once the optimal mapping is found, the distance between the two trajectories is calculated by summing the distance between the matched configurations. An important limitations of this approach is that the trajectories are compared only in workspace.

The motion training system *Just Follow me* [86] (JFM) monitors the user's progress by comparing their motion trajectory to a reference trajectory. JFM finds an optimal mapping between points on the two trajectories before calculating the distance. JFM calculates the optimal mapping by enforcing a chronology criterion on matched configurations and forcing the matched configurations to lie within two seconds of each other.

2.7 Approximating trajectories

My systems for configuration tasks provide restricted feedback. This feedback constrains the user's motions to a predefined set of motion primitives and prompts the user to perform only one of the motion primitives at any given time. These restrictions constraint the trajectories that a user can follow. Therefore, a user can only approximate a trajectory in configuration space with the given motion primitives. Approximating trajectories with restricted motion primitives is an interesting open problem for future systems, which is closely related to the problem of polygonal approximations (PA) for curves.

The input for the PA problem is an ordered set of N points in a plane, S. The objective for the PA problem is to find a set of $k < N - 1$ lines, P, that best approximate the reference curve, S. The most common distance measures between points on S and their corresponding approximating line segment are the vertical or orthogonal distances.

Bellman [9] proposed an optimal algorithm when the reference curve, S, is an analytically described quadratic. Imai and Irai [36] described two variants of the PA problem. The first problem, referred to as min-ϵ, seeks to approximate a set of N points with k straight lines such that the total error of approximation is minimized. The second variant, called min-# , fixes the maximum permissible error, Δ. The goal for the min-# problem is to find the smallest number of line segments that can approximate the data points with approximation error less than Δ.

Dunham [22] solved the min-# problem using dynamic-programming and geometrical arguments. Perez and Vidic [59] gave an optimal solution for the min-ϵ problem using dynamic programming that takes $O(N^2k)$ time. Salotti [69] represented the points in S with vertices of a graph and proposed an A^*-search based algorithm that improved the expected running time to $O(N^2)$. For another error criterion known as the tolerance zone criterion, Chen and Chin [14] proposed algorithms for min-# and min-ϵ problems that run in $O(N^2)$ and $O(N^2 \log N)$ time respectively.

Polygonal approximation is a well-studied problem, with many variants and relaxations. Kolesnikov *et al.* address the PA problem when S is a closed curve. Researchers have studied distance metrics other than vertical and orthogonal distances: city-block metric (Pikaz and Dinstein [61]) and the

least integral squared error (Ray and Ray [66]). Panagiotiakos *et al.* [57] have proposed algorithms for min-ϵ problems extended to higher dimensions.

Some researchers have focussed on approximating curves that are not straight lines. For example, Pavlidis and Horowitz [58] considered the problem of approximating the reference curve with polynomial curves. Drysdale and Strum [21] solve the min-# problem for polygonal curves when the approximating curves are circular arcs and biarcs.

The trajectory approximation problem and the polygonal approximation problem share the goal of approximating a reference curve with simpler curve segments . However, PA and its variants differ from the problem of approximating a trajectory in many respect. For example, none of the approaches above consider restricted control as the approximating curve. Also, when defined in configuration space, the error metrics between the trajectories are not well known. Furthermore, unlike most approaches discussed here, the trajectory approximation problem is not restricted to two dimensions.

Chapter 3

System Hardware

My motion feedback systems interact with several external devices. These devices fall under three major categories: tracking sensors, feedback motors and communication devices. All these devices need to interact with each other in real time for my systems to function properly. Naturally, the performance and limitation of these devices affects the design, performance and limitations of my systems. This chapter briefly describes the working, output, advantages and limitations of the hardware devices I used to build my motion feedback systems. Section 3.1 presents the details of tracking devices. Section 3.2 discusses the feedback motors, the modes they operate in, and different ways to mount them on the user. Finally, section 3.3 describes the communication devices.

3.1 Tracking devices

Tracking devices track the user's current configuration. These devices differ both in technology and their output. Different tracking devices may produce qualitatively different data, and systems often filter and process the sensor data before determining the feedback.

I have used two tracking devices to build my systems: a motion-capture system and a three-axis accelerometer.

3.1.1 Motion-capture system

I used a Vicon MX system running Vicon Nexus software to track users. Vicon MX is an optical motion-capture system that uses multiple high-speed cameras and infra-red light to track users. For tracking a user, retro-reflective markers (figure 3.1) are placed on the user's body. Multiple motion-capture cameras simultaneously capture images of these markers. The tracking software then calculates the position of these markers from these multiple images.

Tracking a user with a motion-capture system involves two steps: defining a tracking skeleton and fitting the tracking skeleton to the marker positions determined by the cameras.

A tracking skeleton specifies the different links in the user's body, the joints that connect those links, and the placement of markers on the links. Vicon Nexus software provides an easy interface for defining tracking skeletons. As the user moves in the capture volume, the motion-capture cameras accurately calculate the positions of the markers mounted on the user's body. However, the markers in the camera images are indistinguishable from each other; the cameras do not know the correspondence between the markers in the images and those defined by the tracking skeleton. Vicon Nexus software attempts to find to find some *best* labeling for the markers based on the skeleton template.

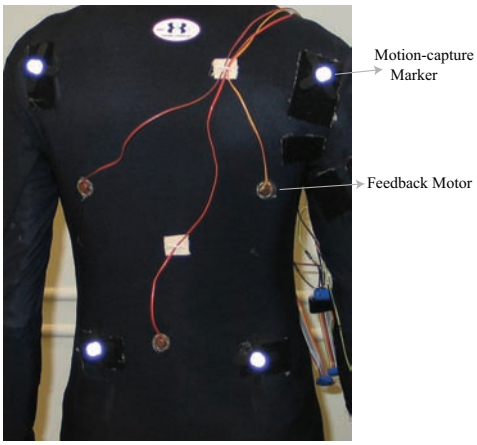


Figure 3.1: A figure showing motion-capture markers for an system that tracks the torso. This figure, also, shows the feedback motors.

The output from the Vicon MX system is a set of marker names and their corresponding 3D positions. Strategically placing the markers, for example on the shoulder, makes it easy to track the position of those body parts. Many applications calculate the configuration space variables, such as joint angles, from the position data before calculating the feedback[1].

Advantages and limitations

A motion-capture system has several advantages for tracking users. The Vicon MX system is extremely accurate (an error of ± 0.1 mm in marker position) and provides high sampling rates of greater than 100 fps. Thus, applications can get accurate position data with low latency. Vicon MX provides an easy, socket-based interface for applications to access this data. Furthermore, the Nexus software allows developers to easily create different tracking skeletons templates. It is not necessary to model the entire body for tracking only certain parts; we only need to create a tracking skeleton template of the relevant parts. Motion-capture systems provide absolute 3D positions of the markers, and so they are especially suitable for applications that need position data such as the mobile manipulation described in chapter 4.

The Vicon MX system has many drawbacks. This system is extremely expensive, and so it is unreasonable to expect that it could be deployed on a mass scale, for example in user's homes. Before using the system, it is necessary to calibrate both the cameras and the user. The calibration step takes between 15 to 45 minutes. Like other optical motion-tracking system, Vicon MX is sensitive to external light sources and reflective materials.

Although the motion-capture systems output marker data in real time, they are not designed for realtime applications. The tracking algorithm frequently mislabels markers and sometimes misses them altogether. Realtime applications, such as the ones that I developed, cannot utilize sophisticated, offline post-processing techniques that solve these problems.

In real-life scenarios, users move in and out of the capture volume. Motion-capture systems that are confined to a room cannot cope with the user's mobility.

Newer camera-based systems such as Microsoft Kinect [42] alleviate some problems associated with marker-based motion capture systems. Kinect is cheaper, does not need retro-reflective mark-

[1]Vicon MX provides the joint angle data, but these angles correspond to a particular convention defined by the system. Knowing the convention, it is possible to calculate the forward kinematics for the system. However, this convention is not intuitive for human feedback, and so I ignored this output.

(a) WiTilt Sensors from Sparkfun Electronics (b) WiTilts mounted on the user's arm.

Figure 3.2: WiTilt accelerometers from Sparkfun electronics. (a) The WiTilt sensors used to measure the acceleration along three mutually orthogonal axes. (b) WiTilts mounted on the user's hand to measure the arm's configuration.

ers, requires little calibration before using it, and is more robust to lighting changes. However, like marker-based systems, it has a limited capture volume.

3.1.2 Accelerometers

Accelerometers measure the acceleration of the device they are mounted on. The source of acceleration may either be the movement of the device or gravity. By measuring the acceleration caused by gravity, it is possible to determine the orientation (tilt) of the device. This method of operation is called the tilt sensing mode for accelerometers.

Modern accelerometers use micro-electromechanical systems (MEMS) to measure acceleration. Tilting the accelerometer moves some micromachined parts in the sensor. This movement produces a change in the electrical properties of the sensor such as its capacitance. The change in the electrical property is proportional to the acceleration, and it is used to calculate the tilt of the accelerometer. Typically, the output from an accelerometer is voltage that is proportional to the acceleration. An ADC converts this voltage to a digital signal which is then read using a microcontroller (please refer to [16] for details of converting accelerometer voltage output to tilt data). Accelerometers differ in the number of axis along which they can measure the acceleration: one-, two-, or three-axis accelerometers.

I used WiTilt v3 [75] modules from Sparkfun electronics (figure 3.2) as my sensors. WiTilts (figure 3.2(a)) contain a three-axis accelerometer (either Freescale Semiconductor Inc.'s MMA7260Q [28] or MMA7361L [29]) as their sensor. These sensors measure acceleration along three mutually orthogonal directions (shown in figure 3.2(a)), and wirelessly transmit the sensor readings over a serial-over-Bluetooth connection using an RN-41 Bluetooth module [68] from Roving Networks. A WiTilt module has its own battery pack that powers the accelerometer and the Bluetooth radio.

WiTilts can stream processed data in multiple formats: acceleration values, voltage values, and tilt values. These formats are ideal for applications that simply measure or store the acceleration data. Alternatively, WiTilts can stream the raw voltage values in a packet format. The packet mode suited my applications better. For using WiTilts in the packet mode, I wrote software to parse the packets, and process the voltage readings to obtain the tilt values. The WiTilt user guide [75] provides formulae to convert voltage data to tilt values.

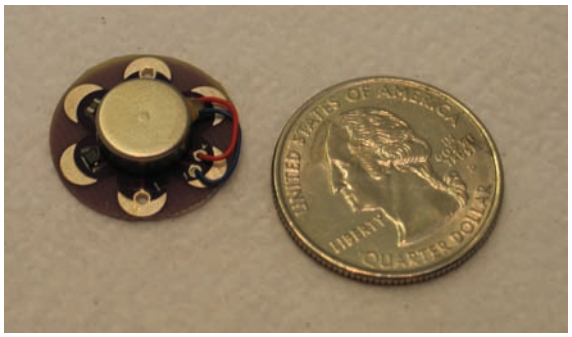


Figure 3.3: Lilypad vibrating motors that act as the feedback device.

Advantages and limitations

WiTilts provide a single package that measures, processes and wirelessly transmits data without needing additional circuitry or power source. Therefore, they are extremely convenient for tracking users. Because they are mounted on the user (see figure 3.2(b)), WiTilts are not confined to a fixed area like optical motion-capture systems. WiTilts (and accelerometers in general) are relatively inexpensive and do not require elaborate setup like motion-capture cameras.

Conversely, WiTilts provide low-resolution and noisier data compared to a motion-capture system. The output from WiTilt does not provide orientation around the vertical axis (because the acceleration is the same for all orientations). Accelerometers are not suitable for applications that require absolute position data such as navigation applications. Other sensors such as compasses or gyroscopes are frequently employed in conjunction with accelerometers to calculate the absolute positions. These units require significant additional processing and are not sufficiently accurate for most applications.

WiTilts require frequent recalibration. Because sensors mounted on body parts are likely to move and slip, every experiment trial must first calibrate the sensors before starting the tracking.

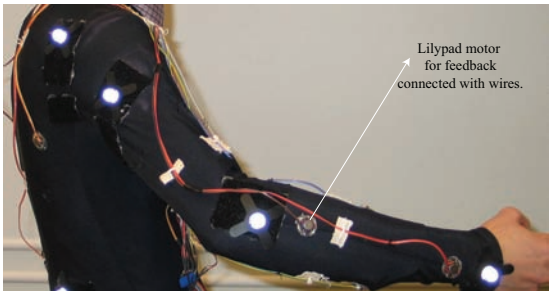
Using WiTilt data required me to write significant code to receive, parse and refresh the data. In streaming mode, special care must be taken to ensure that the application does not use stale data, and the transmitter does not overflow the computer's serial-port buffers. Finally, systems can experience acceleration from sources other than gravity such as user movement, and collision. An accelerometer cannot distinguish these accelerations from the acceleration due to gravity. My applications require users to move slowly, and under those conditions it is reasonable to assume that gravity is the most significant source of acceleration. However, that assumption may not be valid for other applications. In such cases, additional processing is required for accurately tracking the user.

3.2 Feedback motors

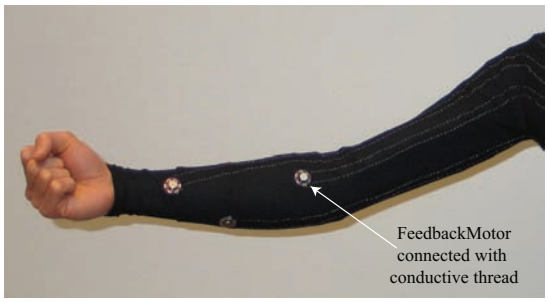
My systems use Lilypad([13]) vibrating motors (figure 3.3) as feedback devices. For my systems, every motor corresponds to a particular motion. These motors are placed at locations that are intuitive for the motion. When a particular motor buzzes, it signals the user to carry out the specified motion.

Modes of feedback

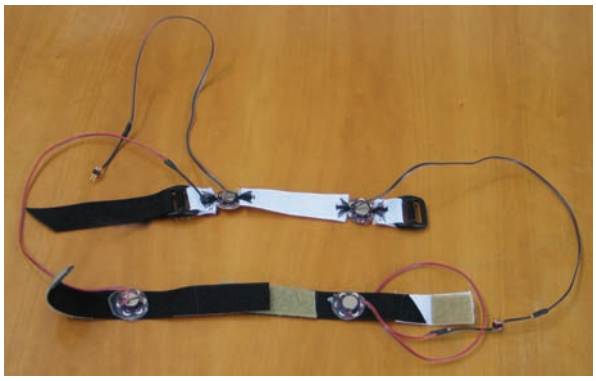
I used the vibration motors in two feedback modes:



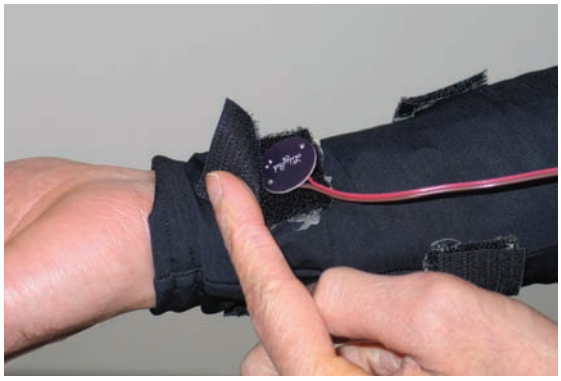
(a) Motors connected with wires



(b) Motors connected with conductive thread



(c) Bands to strap vibration motors



(d) Velcro slots to hold vibration motors

Figure 3.4: Different modes of providing vibration feedback. (a) Vibration motors connected with wires to the electrical input. (b) Vibration motors connected using conductive thread. (c) Bands with vibration motors attached to them. (d) Velcro slots that hold the vibration motors in place.

- **Continuous feedback**: In the continuous mode, a feedback motor buzzes continuously for the duration of the motion. When the controller stops buzzing a motor or switches to a different (set of) motor(s), it signals the user to change their motion.
- **Pulsing feedback**: In pulsing mode, a vibration motor is *pulsed* at fixed intervals. Both the duration of a pulse and the interval between pulses is kept constant.

Continuous mode can only provide binary feedback; either the user is prompted to perform an action or not at all. This feedback mode is easy to implement, because it only involves switching the motor on for the duration of the motion. Continuous mode has low communication overhead for controlling the motors. For every motion, a *high* signal is sent at the beginning and a *low* signal is sent at the end of the motion. Low communication overhead is particularly useful when the feedback software runs synchronously with the tracking and controller software. Fewer packets also imply lower power consumption by the radios.

Pulsing feedback can provide some limited velocity feedback by changing the interval between pulses (see chapter 9 for details of a system that uses this kind of velocity feedback). Pulsing motors requires sending a *high* and a *low* signal at fixed intervals. Therefore, this mode requires two wireless messages every pulse, which greatly increases the transmission overhead and power requirements for the radio compared to continuous mode. For pulsing motors at a regular interval, the feedback software runs asynchronously with the tracking and controller software, either as a separate thread or as a separate process.

	Attached with electric wires	Attached with conductive thread	Bands	Velcro slots
Motor Reusability	Not reusable	Not reusable	Reusable	Reusable
Maintenance	Difficult	Extremely Difficult	Difficult	Simple
Feedback Quality	Excellent	Excellent	Poor	Good
Deployment Time	Negligible	Negligible	15-20 mins	< 5 min
Limitations	None	None	Difficult for Torso	None

Table 3.1: Comparison between different options for mounting feedback motors.

3.2.1 Mounting motors to deliver feedback

Vibration motors need to be mounted on the user's body for providing tactile feedback. I tried the following four strategies for mounting the motors on users (figure 3.4):

- Attached with wires to power the motor: The motors are attached to the user's clothing with glue and connected to the power source with electrical wires (figure 3.4(a)).
- Attached with conductive thread to power the motor: The motors are sown into the user's clothing and connected to the power source using a conductive thread (figure 3.4(b)).
- Bands with attached motors: Motors are attached to bands that can be worn on top of clothing (figure 3.4(c)).
- Velcro slots to attach motors: Velcro patches with holes at their center are attached to the user's clothing. The motors fit in the holes at the center and are held in place by a complementary piece of velcro (figure 3.4(d)).

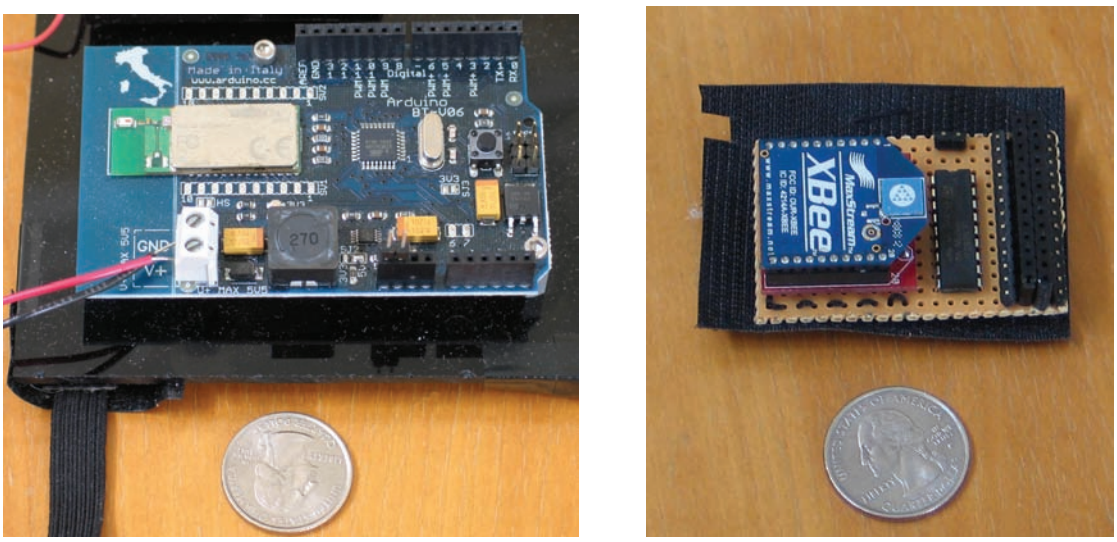
Table 3.1 evaluates each of these mounting strategies for five properties:

- **Reusability**: Ability to use the same motors for different users/systems.
- **Maintenance**: Ease of attaching or detaching the motors in case of a problem.
- **Quality of feedback**: How easy it is to feel and isolate the buzzing from an individual motor?
- **Time of deployment**: The time it takes to mount the motors before using a system.
- **Limitations of the mounting strategy**: Cases where a particular mounting strategy is infeasible.

Mounting strategies that attach the motor directly to the clothing cannot use the same motor across different users or systems. Both bands and velcro patches easily allow reuse of motors across different users/systems.

Velcro patches are the easiest to maintain because the motors are never attached to any external surface. Attaching motors with conductive thread fares the worst, because to replace a motor it must be sown again from the mounting position to the power source. Additionally, conductive thread suffered significant problems with loose contacts.

Not surprisingly, systems that have motors attached directly to the clothes provide the best quality of feedback in terms of identifying and localizing the motor's vibration. Feedback bands



(a) Arduino BT microcontroller board.

(b) XBee RF radios.

Figure 3.5: Two communications devices I used for my systems. (a) An Arduino BT microcontroller board. (b) An XBee RF radio.

provided the worst quality of feedback. The bands absorbed and diffused the motor's vibration, making it difficult to feel and localize the vibration.

All systems except bands had low setup time. Setup time was least when the motors were directly attached to the clothes. Finally, feedback bands faced an additional limitation because they cannot be easily deployed to all parts of the body, *e.g.*, the torso.

3.3 Communication devices

My applications switch feedback motors on/off depending on the controller's feedback. Communication devices act as a bridge between the controller and the feedback motors. On one end, communication devices receive instructions from the controller regarding motors to switch on/off. On the other end, they control the power to the feedback motors using their I/O pins. When the controller sends instructions to switch on(off) a particular motor, the communication devices turn on(off) the power to that particular motor.

Communication devices are mounted on the user, and for my systems, communicate with the controller over a Bluetooth or wireless connection. I have used two communication devices for my applications: an Arduino BT microcontroller [3] and XBee RF modules [20]. In the rest of this section, I will present the details of these devices, the extra circuitry needed to connect them to the motors, and compare these two devices.

3.3.1 Arduino microcontroller boards

Arduino BT (figure 3.5(a)) is an microcontroller board based on the ATMega168 [4] microcontroller from Atmel systems. It has a Bluegiga WT11 [10] Bluetooth module that allows it to communicate with a serial-over-Bluetooth connection. There are 14 digital I/O pins (eight digital and 6 analog+digital pins) on the Arduino BT board. My systems used these I/O pins to drive feedback motors. I used a battery pack with four rechargeable AA batteries to power the microcontroller

board.

The microcontroller board is programmed using the Arduino software. My systems control feedback motors by sending data instruction to the Arduino BT module. A program running on the microcontroller board constantly checks for input from the controller. In response to the controller's instructions, the program running on Arduino BT changes the state of its I/O pins, controlling the feedback motors.

There is no predefined packet structure for communicating with Arduino BT. A simple user-defined protocol is sufficient to control motors connected to the microcontroller board. The ATMega processor on Arduino BT provides 16 KB of flash memory for storing application programs. Therefore, Arduino BT may be used for more complex applications than simple I/O line passing.

3.3.2 XBee RF modules

An XBee module (figure 3.5(b)) from Digi corporation [20] has an RF antenna that communicates over a serial interface. These modules have nine digital output pins (eight are I/O while one is output only) that may be used to control feedback motors. The XBee interface provides functions to control its I/O pins. Therefore, XBees are widely used to add wireless capability to systems that need to interact with external sensors.

Interfacing with XBee modules and using them to control feedback motors is more complex than using Arduino BT microcontrollers. Unlike Arduino BT boards, XBees are not programmable; the firmware runs on the module's EEPROM and provides a fixed interface to control or configure the I/O pins. Every XBee module has a unique 64-bit address, and allows only one other XBee module to control the state of its I/O pins[2]. This restriction on communication implies that in addition to the remote XBees connected to the motors, my systems needed a base module to relay the controller's instructions. The maximum current rating for XBee pins (2mA for input voltage between 2.8V–3.3V) is too low to directly drive vibration motors. Therefore, XBee pin's output acts as the input to a transistor switch that controls the feedback motors.

Figure 3.6 shows the schematic for using XBee modules to control the feedback motors. The controllers for my systems run on a computer. They send their instructions to the base XBee module. The base XBee module runs in a transparent mode, *i.e.*, it transmits the instructions from the controller over the wireless antenna. These instructions are captured by the remote XBee module that is configured to accept I/O instructions from the base module. In response to the instructions, the remote XBee module changes the output state of its I/O pins. The I/O pins are connected to a transistor array, which act as switches for the vibration motors. I used the ULN2803A [79] chip from Texas Instruments. This chip has an array of eight-NPN transistors that work in the common emmiter configuration.

Comparison between Arduino BT microcontrollers and XBee modules

Table 3.2 compares the two communication devices. The XBee modules are much smaller and cheaper that Arduino BT boards. On the other hand, Arduino BT boards are easier to use: they need no external electronics, allow simple communication protocols and allow user programs to run on the microcontroller board. Additionally, the Arduino BT boards provide rich functions such as clock timers, libraries for reading analog or I2C [56] inputs, and reference voltages to drive external devices. Therefore, it is much easier to, for example, pulse motors using an Arduino BT board.

[2]XBees have an *open* mode where they allow any XBee module to change the state of their I/O pins. However, this open mode may cause changes in the state because of stray packets. The state changing packet must originate from another XBee module.

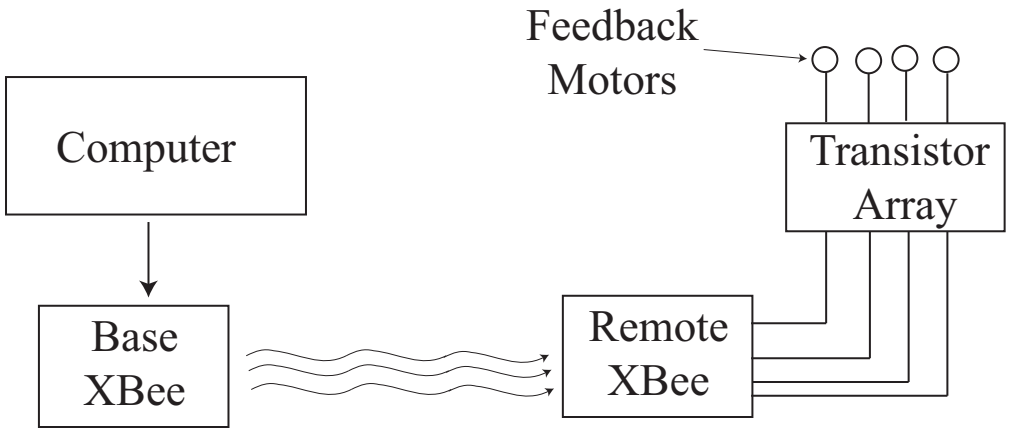


Figure 3.6: A schematic diagram showing the working of the XBee module. The computer sends a message to the base XBee module, which retransmits those messages to the remote module. The remote modules changes its pin output based on the message it receives. A transistor array acts as a switch that drives the motors in response to the output of the XBee pins.

	Arduino BT	XBee
Footprint	2.1in × 3.2in	1.087 in × 0.96 in
Cost	$150	$20
Number of available pins	10	9
Extra electronics required	None	transistor array and connectors
User programs	Allowed	Not allowed
Communication Protocol	None	Predefined packet structure

Table 3.2: Comparison between Arduino BT and XBee.

However, for the simple purpose of controlling motors, XBee proved adequate and cheaper. Their smaller footprint is a significant advantage; these communication devices have to be mounted on user's body where space is scarce. Most of my applications preferred XBee modules over Arduino BT microcontroller boards because of their smaller footprint

Chapter 4

Mobile Manipulation System

The mobile manipulation system allows a blindfolded user to navigate a room and manipulate objects. Such a system could be used as an aid to the visually impaired or provide task-based motion training. For example, in chapter 6 I will present a system that guides a user to fold a T-shirt by extending the manipulation system from this chapter.

The input to the mobile manipulation system is a sequence of navigation and manipulation tasks. Each task specifies its target configuration, and the system guides the user to sequentially achieve those target configurations.

4.1 System design

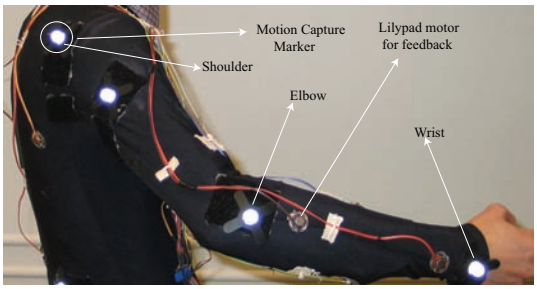
My system uses a motion capture system to track the user's current configuration and Lilypad vibration motors to guide the user to the target configuration. The motion capture system tracks the user's current position, orientation, and the configuration of his arm.

The mobile manipulation system consists of two subsystems: the manipulation system and the navigation system. The manipulation system takes the target location (in world coordinates) as input, and guides the user's hand to the specified location. The navigation system guides the user to a target location and orientation, where orientation refers to the direction that the user is facing.

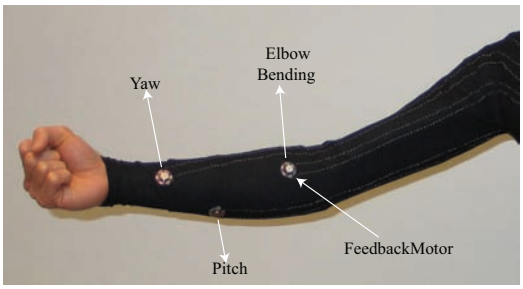
For both the navigation and manipulation subsystem, my system describes a set of motion primitives. Given a task, these systems prompt the user through a set of motion primitives that guide him to the target configuration. Task-based systems use restricted feedback to guide the users. I will first describe the restriction on feedback in section 4.1.1, followed by the details of the manipulation (section 4.1.2) and navigation systems (section 4.1.3).

4.1.1 Restricted feedback

Planning tasks for humans is fundamentally different from planning tasks for robots. Reaching a doorknob is a good example to illustrate this difference. To reach the doorknob, consider a planner that calculates the initial and final configurations of a robot and interpolates a path between them. Such a planner typically expects precise compliance with feedback across multiple degrees of freedom, and a low-latency control loop. This planning strategy is not ideal for humans for many reasons. First, human response to feedback is error-prone and unpredictable. Next, the lag between a controller command and the response to it is typically orders of magnitude higher in humans than in machines. Finally, communicating feedback for multiple degrees of freedom is straightforward for robots. On the other hand, it is hard to convey such feedback to humans, in a



(a) Manipulation System Markers



(b) Manipulation System Motors

Figure 4.1: (a)Marker positions for the arm skeleton used by the posing system. (b) The placement of Lilypad motors on the arm for the manipulation system. Each of the motors shown here has a corresponding counterpart (not visible in this figure) to prompt the user to move in the "opposite" direction.

manner that the user finds easy to interpret and act on. To counter these human limitations, the systems for configuration tasks (described in chapters 4, 5 and 6) place the following restrictions on the feedback:

- **Restricted control**: The user is restricted to a discrete set of predefined actions, the *motion primitives*. These motions are *natural* – they do not need complex training – and they depend on the task.
- **Binary feedback**: Either the user is prompted to perform a certain action, or not at all. Our system does not try to control the speed of the motion.

The mobile manipulation system provides feedback using Lilypad [13] vibration motors. Two vibrating motors are typically used for each motion primitive. The motors are placed at locations that are intuitive for the particular action. For example, if the motion primitive is bending the elbow, a motor is placed near the elbow. When a motor buzzes, it signals the user to carry out the associated motion primitive. Restricted control implies that at any given time the system buzzes only one motor.

The primary advantage of restricted feedback is the simplicity of determining and interpreting feedback. A limited set of motion primitives of which only one is prompted at any given time aids the user in interpreting and complying with the motion feedback. Conversely, it is not possible to provide feedback for trajectory-based tasks using restricted feedback.

4.1.2 Manipulation System

The manipulation system takes the target location (in the world coordinate frame) as its input, and guides the user's hand to the specified location. The motion-capture system tracks the positions of four markers placed on the user's arm (figure 4.1(a)), and the user's shoulder acts as the origin of the body frame of reference for that arm. The configuration space variables for the system are the spherical polar coordinates for the hand, (r, θ, φ), where r is the distance from the origin, θ is the longitude, and φ is the colatitude.

The motion primitives for the manipulation subsystems are three free rotations around joints: yaw and pitch rotation around the shoulder joint and the bending of the elbow joint (figure 4.2). Changing the yaw and pitch values of the shoulder joint changes the longitude and the colatitude (θ and φ) of the hand by the same amount (approximating the arm as a rigid body). Bending the elbow changes the distance (r) of the hand from the origin (the shoulder). Thus, using these motion

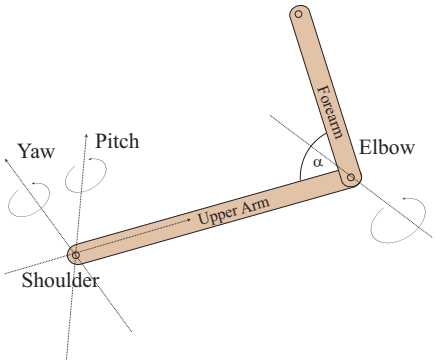


Figure 4.2: The motion primitives for the manipulation system. The motion consists of the yaw and pitch rotation around the shoulder joint, and the bend at the elbow joint.

primitives, the controller can independently change each of the configuration space variables. The motion primitives correspond to natural rotations around the joints, and so they are easy to follow. The mobile manipulation system relies on the human being for the complex task of sensing and grasping the object properly.

It is easy to verify that sequentially correcting (changing the variable till it matches the target value) each degree of freedom will guide the hand to the target location. Precise compliance is difficult for humans. Therefore, the controller changes the correcting degree of freedom based on two criterion: either the difference in the current and target value of the corrected degree of freedom is less than a threshold, or the user has overshot the target value. If after applying all three motion primitives, the arm is farther from the target than a specified threshold, the controller repeats the sequence of motion primitives till the arm reaches the target location. I have tested two criterion for selecting the next motion primitives: round-robin and maximum error. Both lead to quick convergence towards the target location.

4.1.3 Navigation system

The navigation system guides the user to a target location and orientation, where orientation refers to the direction that the user is facing. This system allows a user to navigate an obstacle-free room, and could potentially be used as an indoor navigational aid for the visually impaired.

The navigation subsystem tracks the user with a set of four markers placed on his back. These markers follow the position of the left and right shoulder and the left and right extremities of the waist just above the pelvis. The motion-capture markers and feedback motors are placed on the user's back, as shown in figure 4.3.

My system estimates the user's position as the vector average of the four marker positions (ignoring the z component), and his orientation as the normal to the plane defined by the markers. In three dimensions, three non-collinear points define a plane. Consequently, any three markers on the back define a plane, and hence an orientation for the actor. With four markers on the back, there are four candidate orientation vectors($^4C_3$) for the user. The current implementation estimates the actor's orientation as the average over these four orientation values.

The navigation system utilizes two natural motion primitives: walk straight ahead, and turn in place. The walk-straight primitive needs one motor that prompts the user to walk straight ahead. The navigation system uses two motors for the turn-in-place primitive: one to indicate turning left and another to indicate turning right.

The controller uses a simple "turn-drive-turn" algorithm. In the fist phase, the controller orients

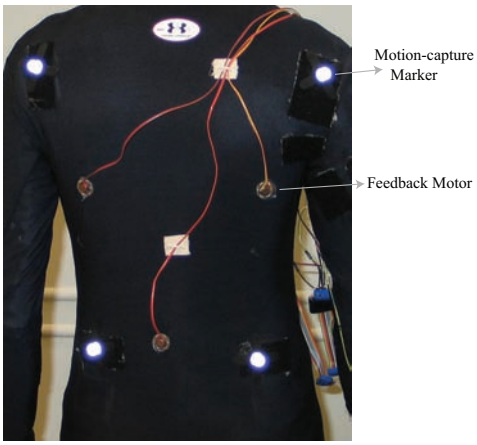


Figure 4.3: The placement of the markers and motors for the navigation system. There are four markers two on each shoulder and two on the waist just above the pelvis. The three motors prompt the user to move forward, turn left, or turn right.

the user towards the target location, and then prompts him to walk straight towards it. Since the amount of turning and walking straight might not be precise, this phase may need multiple iterations till the user reaches the target location. Once at the target location, the user is prompted to turn in place until he is aligned with the target orientation.

4.2 Testing and future work

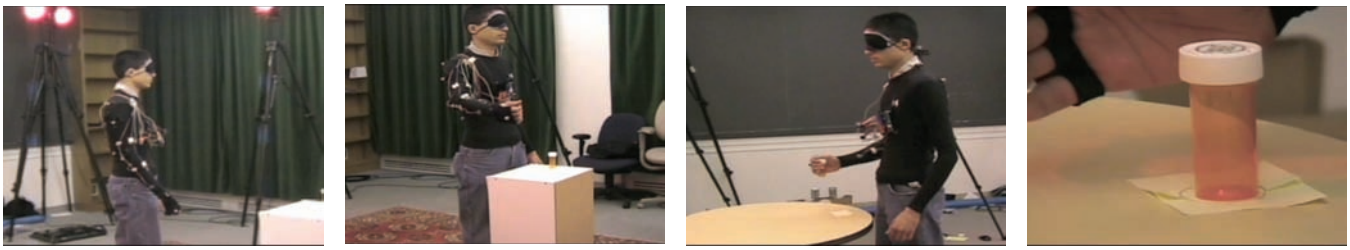
I tested both the manipulation and navigation subsystems separately before testing the mobile manipulation system. I tried a range of destinations for the arm's target position for testing the manipulation system. Similarly, I tried many start and target orientations for testing the navigation system. References [47] and [54] show videos of one of my test runs. The mobile manipulation system can direct a user through a sequence of manipulation and navigation tasks. Figure 4.4 shows snapshots (from [53]) of the user performing different tasks in a sequence. From an arbitrary start location, the user was prompted to navigate to a particular location, and pick an object placed nearby. Then, the controller guided him to a new location to place the object at a nearby target location. The video clip for this experiment is provided in reference [53]. I repeated this experiment multiple times with different targets for the manipulation and navigation tasks.

In my experiments the navigation system could guide the user to within 15 cm of the target location and within 15° of the specified orientation. The manipulation system could place the object to within 1 cm of the target location. In some cases, both the manipulation and navigation systems required multiple iterations for correcting the same degree of freedom to achieve the desired accuracy.

The mobile manipulation system (and the systems developed in chapters 5 and 6) were preliminary systems that were developed for testing the feasibility of providing real-time feedback for human motion tasks. I did not conduct user tests for these systems. Later systems (described in chapters 7, 8, and 9) have detailed user-testing results.

Limitations and future work

My experiments validate two design decision for the current system: the principle of restricted feedback and using human sensing and manipulation abilities. These design choices enable my



a b c d

Figure 4.4: Different phases of the mobile manipulation application. (a)The blindfolded user is prompted to walk to a location. (b) The user picks an object placed nearby. (c) The user is prompted to walk to a new location. (d) The user is guided to place the object at a specified location.

system to complete interesting tasks with minimal tracking and feedback. However, the current system has limitation that future system may address.

The motion capture system is expensive and confined to a room. Therefore, applications targeted at the mass market that use a motion capture system may not be viable. Other tracking systems such as Microsoft's Kinect [42] or North Star from Evolution robotics [55] in conjunction with other cheap sensors, such as accelerometers and gyroscopes, may make such application more accessible to the general public.

Restricted feedback limits the present setup to task-based systems because the path followed to achieve the target configuration is not controlled. However, trajectory objectives require the user to follow a particular trajectory in configuration space. It may not be possible to decompose such a trajectory into a sequence of motion primitives.

The results of this experiment show that the present setup can be applied to other applications. The objectives for the mobile manipulation systems are what could be described as point objectives: the system's objective is to place one particular point on the user's body, such as the hand, at a target location and orientation. I have extended this system to pose objectives (chapter 5), where the goal configuration is specified for the entire system instead of for a single point. Chapter 6presents the details of an application that extends the manipulation system to both arms and helps a user fold a T-shirt.

4.3 Lessons learned

The mobile manipulation guidance system had to simultaneously interact with multiple components such as the motion capture system, microcontroller board, etc. One of the limiting factors about testing such a system is the setup time. Calibrating motion capture cameras, user skeletons, establishing communication with the microcontroller and testing motors took 30-45 minutes. Unfortunately, this effort needed to be repeated every session.

Realtime computations on the tracking data made debugging this application extremely challenging. I wrote a simulator to minimize this problem. The simulator simulated the motion of the arm and the body in response to the feedback provided. It also plotted the user's configuration using a basic GUI. The simulator enabled me to correct errors in the controller's logic without using the motion capture system. I used the simulator for all systems designed for configuration tasks.

Motion capture systems are primarily designed for offline analysis of the captured motion data. Mislabeling or losing markers sometimes caused the controller algorithm to produce erroneous feedback. A related problem concerns the difference between the position of a marker and the

point on the body the marker represents. This error is difficult to model or predict because of the flexible nature of the human body.

Chapter 5

Arm Posing System

A pose specifies the configuration of all joint angles that define the system. Many real-world applications require users to assume poses. For example, in dance and martial arts taking a certain pose is an important step that requires precise bending of body segments. In sign language a pose implies a specific word. Many exercises require the actor to assume a pose before starting the routine.

The ability to assume whole body configurations may add redundancy to a system which could be used to avoid obstacles in the workspace. For example, while positioning the hand at a target location, it may be necessary to assume a pose such that the elbow does not collide with obstacles.

5.1 System design

I developed a system that guides a four degree-of-freedom model of the arm to a specified pose. The human arm has seven degrees of freedom: three in the shoulder, one in the elbow, one in the forearm, and two in the wrist. My system ignores the two degrees of freedom in the wrist and one in the forearm. For the arm-posing system, a pose is specified with four joint angles: the yaw (θ), pitch (φ) and roll (β) for the shoulder joint, and the elbow bend (α).

The input to the posing system is the target pose defined by a set of four joint angles. I used a motion capture system to track the user. For this system, I used a four marker motion-capture skeleton. Figure 5.1 shows the placement of these markers. The motion capture system tracks the set of four markers and returns the positions of the shoulder, elbow and wrist. Given these marker

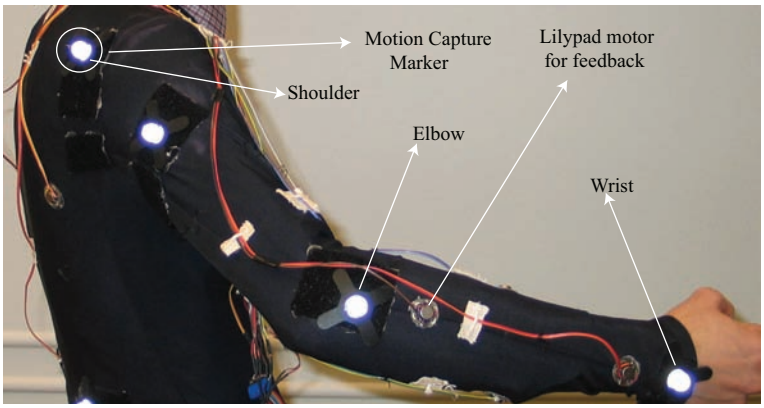


Figure 5.1: Marker positions for the arm skeleton used by the posing system.

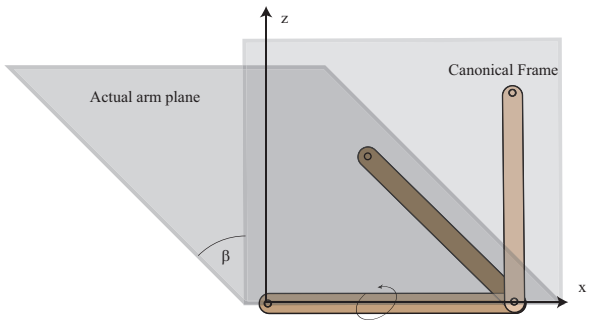


Figure 5.2: The"roll" degree of freedom is the rotation of the arm when the axis of rotation lies along the upper arm. This motion rotates the plane in which the shoulder, elbow and wrist lie. The angle between the canonical plane and the actual arm planes, β, is the roll in the arm.

positions the controller calculates the joint angles by solving the inverse kinematics for the arm. Based on the current and the target joint angles, the posing system directs the user through a set of motion primitives to achieve the target pose.

Calculating the user's pose from marker data

In this section, I will assume that the user's local frame corresponds exactly to the global frame of the motion capture system. In the user's local frame, the z axis corresponds to the vertical direction, the y axis corresponds to the direction the user is facing, and the x axis corresponds to the sideways direction. The origin of the user's frame lies at the base of the shoulder.

For any non-singular position (when the arm is not straight), the controller uses the canonical and actual position of the arm-plane (plane defined by the shoulder, elbow and wrist see figure 5.2) to determine the roll angle. In its canonical position the arm-plane is coplanar with the x-y plane, and the upper arm points along the x axis. With this definition, the yaw, pitch and roll degrees of freedom are rotations around the z, y, and x axes respectively.

Starting with the arm in the canonical plane, any pose given by $(\theta, \varphi, \beta, \alpha)$ can be attained with the following sequence of motions:

- Extend the arm to reach the correct α value.
- Rigidly rotate the arm along the x, y and z axes respectively until the arm reaches the target values.

Importantly, the last three rotations rotate both the elbow and the wrist by the same amount. Calculating the yaw and pitch values from the marker positions involves a simple conversion from Cartesian to spherical polar coordinates. Similarly, the bend in the elbow can be determined using the law of cosines on the triangle defined by the shoulder, elbow, and the wrist. These calculation are the same as that for the mobile manipulation system in chapter 4. The rest of this section provides the details for calculating the roll angle.

Let the elbow's position be $X_1 = (x_1, y_1, z_1)^T$ and the wrist's position be given by $X_2 = (x_2, y_2, z_2)^T$. Further, let the wrist's position in the arm's canonical position after extending the arm to the correct α value be $X_2^c = (x_2^c, y_2^c, z_2^c)^T$

The rotation matrices for rotations around the x, y and z axes are the well-known rotation matrices for rotation around the three Cartesian axes. Let these matrices be called R_x, R_y and R_z. For a given pose $(\theta, \varphi, \beta, \alpha)$, the rotation around the z and y axis are θ and $(\pi/2 - \varphi)$. Transforming the wrist from the canonical location (X_2^c) to the target position (X_2) can be expressed as a sequence of three rotations:

$$X_2 \;=\; R_z(\theta) R_y(\pi/2 - \varphi) R_x(\beta) X_2^c \; . \tag{5.1}$$

For the sake of readability, I will make the following substitutions:

$$\begin{array}{ll} c_\beta = \cos\beta & s_\beta = \sin\beta \\ c_2 = \cos\varphi & s_2 = \sin\varphi \\ c_3 = \cos\theta & s_3 = \sin\theta \end{array} \; .$$

With these substitutions, equation 5.1 can be written as:

$$\begin{pmatrix} c_3 s_2 & -s_3 c_\beta + c_3 c_2 s_\beta & -s_3 s_\beta - c_3 c_2 c_\beta \\ s_3 s_2 & c_3 c_\beta + s_3 c_2 s_\beta & c_3 s_\beta - s_3 c_2 c_\beta \\ c_2 & -s_2 s_\beta & s_2 c_\beta \end{pmatrix} \begin{pmatrix} x_2^c \\ y_2^c \\ z_2^c \end{pmatrix} = \begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} \; .$$

The above equation gives us three linear equations that express $\cos\beta$ and $\sin\beta$ in terms of known values. The solution for the roll angle is as follows:

$$\left.\begin{array}{rcl} c_\beta &=& \frac{k_4 \cdot x_2' - k_2 \cdot y_2'}{k_1 \cdot k_4 - k_3 \cdot k_2} \\ s_\beta &=& \frac{k_1 \cdot y_2' - k_3 \cdot x_2'}{k_1 \cdot k_4 - k_2 \cdot k_3} \end{array}\right\} \varphi \neq \pi/2 \tag{5.2}$$

$$\tag{5.3}$$

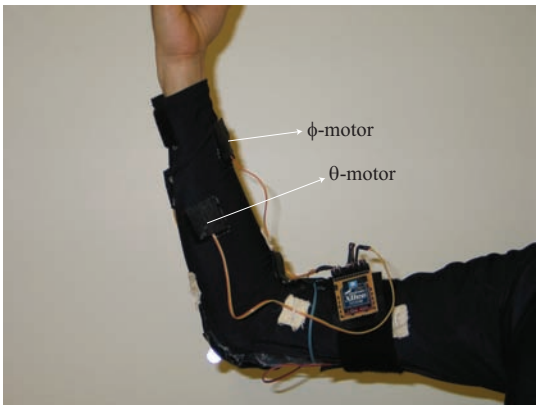
$$\left.\begin{array}{rcl} s_\beta &=& \frac{x2}{k2} \\ c_\beta &=& \frac{z_2^c}{z_2} \end{array}\right\} \varphi = \pi/2, k_2 \neq 0 \tag{5.4}$$

$$\tag{5.5}$$

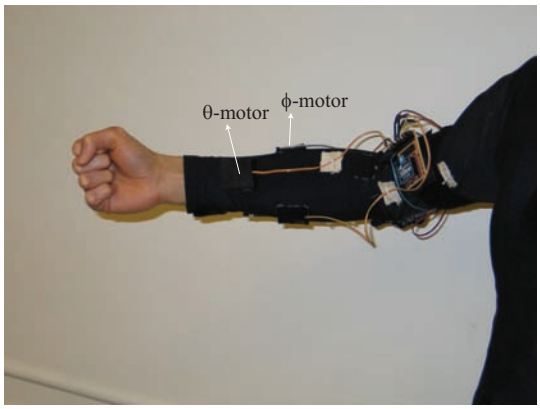
$$\left.\begin{array}{rcl} s_\beta &=& y_2'/k_4 \\ c_\beta &=& z_2^c/z_2 \end{array}\right\} \varphi = \pi/2, k_2 = 0 \tag{5.6}$$

where

$$\begin{array}{rcl} k_1 &=& -s_3 \cdot y_2^c - c_3 \cdot c_2 \cdot z_2 \; , \\ k_2 &=& -s_3 \cdot z_2^c + c_3 \cdot c_2 \cdot y_2^c \; , \\ k_3 &=& c_3 \cdot y_2^c - s_3 \cdot c_2 \cdot z_3^c \; , \\ k_4 &=& c_3 \cdot z_2^c + s_3 \cdot c_2 \cdot y_2^c \; , \\ x_2' &=& x_2 - c_3 \cdot s_2 \cdot x_2^c \; , \\ y_2' &=& y_1 - s_2 \cdot s_3 \cdot x_2^c \; . \end{array}$$



(a) Manipulation System Markers



(b) Manipulation System Motors

Figure 5.3: The feedback motors are placed at intuitive locations for a motion primitive. However, the intuitive mapping between a motion-primitive and the feedback motor depends on the configuration of the arm. The two figures show that the motors that signal the yaw (θ) and pitch (φ) motion-primitive are interchanged for the two configurations of the arm.

Given c_β and s_β, I used the two-argument arc tangent function to calculate the roll angle:

$$\beta = \text{atan2}(s_\beta, c_\beta) \ . \tag{5.7}$$

Controller algorithm

The motion primitives for this system are three natural rotations around the shoulder joint and the bending of the elbow. Each of these motion primitives changes the corresponding degree of freedom for the arm.

The posing-system controller obeys the restrictions on feedback described in chapter 4. At any instant, the arm-posing system prompts the user to perform only one motion from the set of motion primitives. The controller sequentially corrects one degree of freedom at a time by prompting the appropriate motion primitive. With precise compliance, the controller needs to correct every degree of freedom only once. Since precise compliance is difficult, the controller relies on the same two criterion that the mobile manipulation system controller relies on for switching from one control to the next: either the difference between the current value and target for the present degree of freedom is less than a threshold, or the user overshoots the target value. I have tried two criterion for selecting the next control: round-robin and maximum error. Round-robin criterion selects the next control in predefined sequence. This criterion is easier to learn, because the user can *expect* the next correction angle (if the user knows the control sequence). The maximum-error criterion calculates the difference between the user's state and the target pose for all degrees of freedom, the *error* for that degree of freedom. The controllers selects the degree of freedom with the maximum error and provides feedback to correct it. Both these criterion provided quick convergence to the target configuration. The choice of the correct criterion depends on the application.

My systems place feedback motors at intuitive locations for a particular motion primitive. These motors *push* the user in the corrected direction. However, the mapping between a motion primitive and the motor that signals it depends on the arm's configuration. Figure 5.3 illustrates this problem of non-static mapping for the yaw and pitch degrees of freedom. The motors that intuitively signal yaw and pitch motions are exchanged for the two arm configurations shown in

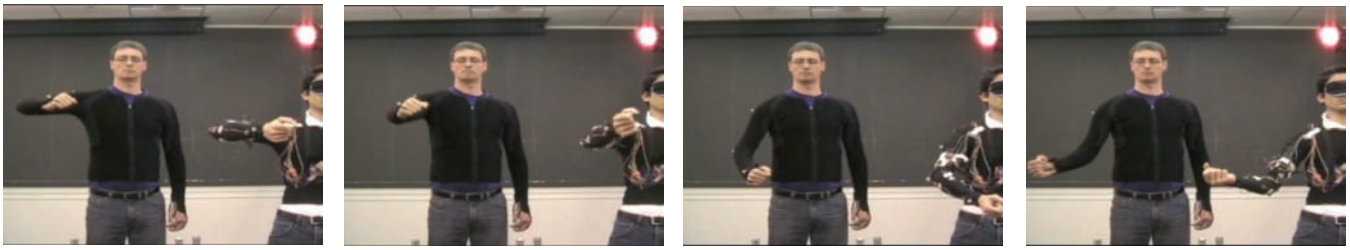


Figure 5.4: The posing system guides the blindfolded user to copy the poses assumed by the second user. The actor on the left assumes a pose, and the blindfolded actor on the right is prompted to copy it. These images from ([18]) shows a set of four successive poses my system allowed the user to copy.

figures 5.3(a) and 5.3(b). The correct mapping depends on the roll value of the arm. Before providing feedback, the controller reevaluates the motion-primitive-to-motor mapping based on the arm's current configuration.

5.2 Testing and future work

The posing system tracked the user's arm with four markers placed on the user's arms as shown in figure 5.1. The motion capture system returns the position of these markers that is used to calculate the user's pose. Figure 5.3 shows the position of the motors and their respective motion control. The arm posing system uses the user's current roll angle (β) to solve the problem of ambiguous motor to motion-primitive mapping described in section 5.1. The controller changes the mapping depending on whether the roll angle is less than 30°, between 30° and 150°, or greater than 150°.

I developed a system that lets one user copy poses assumed by another user. In this system, the first user assumed a pose unknown to the second (blindfolded) user, and the arm-posing system guided the second user to that pose. The motion capture system was used to track both the users. Figure 5.4 shows a set of images of successfully completed poses from the video in [18]. These figures show the ability of my system to accurately reproduce a pose, and the potential use for such a system as a training tool for more complex motions.

We further demonstrated the utility of the posing system by guiding a user to pick an object out of a box. Reference [64] shows a video of this application. This task is not possible with the manipulation system in chapter 4, because for some arm configurations the sides of the box will prevent the user's hand from entering the box.

Limitations and future work

My system set an error threshold of 10° in each joint angle; at the termination of the pose the user's joint angles were within 10° of the target pose configuration. An error threshold recognizes the limitations of the users and the motion capture system. There are many sources of errors such as slipping of markers, small difference in marker placement on the user, human limitations, etc. Some of these errors such as exact positioning of markers may be reduced further with extremely careful calibration. Other sources of errors such as unreliability in human compliance are difficult to measure and reduce.

Future applications may use alternative sensors for tracking the arm poses. Accelerometers, compasses, gyroscopes, or IMUs can potentially replace the motion capture system. These sensors will also eliminate the need to confine the posing system to the capture area.

Finally, my system can be extended to other parts of the body such as legs, torso, or even to whole body systems. Many applications such as yoga, dance, and martial arts require the user to maintain poses and may benefit from such extensions.

5.3 Lessons learned

The motion capture could not reliably track two users in the capture volume, especially when the users were close to each other. It was difficult to consistently test the copying-poses application for one-arm poses, because of unreliable tracking by the motion capture system. However, this problem became worse when I tried building a system for copying two-arm poses. The controller could guide both arms to a specified two-arm pose, when only one user was present in the capture volume. However, I could not test the two-arm copying-poses system, because the motion capture system could not consistently track two users.

In figure 5.4, the palms of the two users are facing in different directions. This difference arises because the users have different twists in their forearms; our system does not control this degree of freedom. Although not a limitation of the posing system, such differences can be disconcerting. Developing the posing system for many degrees of freedom system may solve this problem. However, such a solution would require placing more markers, motors, and communication devices on an already-crowded arm.

The intuitive mapping from motors to motion primitives changes with the arm's pose. My controller changed this mapping based on the arm's current orientation. This adaptive mapping made the system more intuitive.

Chapter 6

Shirt Folding System

I have developed a system that guides a blindfolded user to fold a T-shirt. This system shares many characteristics of configuration-task systems described in chapters 4 and 5. The shirt-folding system uses a motion-capture system to track the user, and Lilypad vibration motors [13] as feedback devices.

On the other hand, this system differs from previous systems in many aspects. The shirt-folding system is more complex than other systems, and controls two arms using restricted feedback. The motion primitives for arm movement are different from those used in mobile manipulation (section 6.1 explains the new motion primitives). The shirt-folding system combines some basic motion primitive into high-level motion tasks, and uses a sequence of such motion tasks for folding a T-shirt.

Another major area of difference is the nature and timing of defining the tasks for the system. T-shirt folding requires several steps. The initial configuration for any folding step (except the first step) is the target configuration for the previous step. The target configurations are specified relative to the initial configuration. From the controller's perspective, the folding strategy is a hybrid of open and closed loop control. The steps undertaken to fold a T-shirt are defined in an open-loop manner, but the actual target configurations that achieve the fold are specified at run time.

Perhaps the biggest contribution of the shirt folding system is that it demonstrates the ability of my system-setup to perform complex tasks. Folding a T-shirt requires the user to perform a long sequence of motion primitives, where errors may accumulate.

The rest of this chapter describes the design and implementation of the shirt folding system. Section 6.1 describes the system's design: the motion primitives, the tracking and feedback systems, and the different steps involved in folding a T-shirt. Section 6.2 presents the experimental results, limitations and future work. I will conclude with some lessons learned while designing and implementing this system in section 6.3.

6.1 System design

This section describes the design of the shirt-folding system. The shirt-folding system's design has three major components: the motion primitives used for folding a T-shirt, the tracking and feedback system, and the sequence of steps that my system uses to fold a T-shirt. The rest of this section presents the details of these three components.

Motion primitives

The shirt folding system uses the Cartesian coordinates of one fingertip on each arm to represent the system's state. Tracking two fingertips implies that the system has six degrees of freedom, three for each fingertip. The feedback motors are placed on the user's upper arm and forearm.

I have made a simplifying assumption about the user's orientation with respect to the world coordinate frame: the user stands facing the positive y coordinate axis. In this orientation, motion along the forward/backward direction corresponds to moving along the y axis. Similarly, moving the arm sideways corresponds to moving it along the x axis. The z axis corresponds to the vertical direction. Moving the arm up or down corresponds to motion along the z-axis. This assumption about the user's orientation is not strictly necessary. The user's orientation with respect to the world frame is easy to calculate, and converting the gripping finger's global coordinate to its local coordinates (in the body frame) corresponds to a simple multiplication with a rotation matrix. Figure 6.2 shows the canonical directions for one arm.

The motion primitives correspond to moving the arm along the three canonical (x, y, z) axes. These motions corresponds to moving the arm sideways (left/right), forwards/backwards, and up/down respectively. All these motions are natural for humans, and need no special learning. There are six motion primitives for each arm (moving the finger along positive and negative directions for the three axes) for a total of 12 primitive motions.

I have defined a set of high-level motion tasks by combining these simple motions. Each high-level motion task involves moving the arm(s) to follow a particular sequence of motion primitives. Some of these high-level motion tasks involve moving both arms simultaneously. Moving two arms violates the strict definition of restricted feedback. However, either arm follows only one motion primitive at a time. Each step in the shirt-folding process is achieved using one high-level motion task. The high-level motion tasks are as follows:

- **Place one arm:** This task moves one hand to a particular location while the other arm remains fixed. The placing algorithm executes a sequence of motions, each involving moving the hand parallel to one of x, y, or z axes. The motion along an axis continues till the coordinate value for the hand matches the corresponding coordinate value for the target location. The controller then corrects another coordinate value, and continues correcting one coordinate at a time until the hand reaches the target location. I have tried two control strategies to select the *next* coordinate to correct: round-robin and maximum error. The round-robin strategy corrects the three coordinate values in a fixed sequence. My system allows the user to set a specific round-robin sequence. The maximum-error strategy always corrects the coordinate with maximum error first.

- **Move arms parallel:** This task simultaneously moves both arms parallel to one of the three canonical direction and in the same direction. For example, both arms move up (parallel to z axis) till they reach their respective target height.

- **Lay-down forward:** One common shirt-folding maneuver involves spreading out a portion (or whole) of the T-shirt on the table while lowering it. The shirt-folding system achieves this task by lowering the T-shirt in a ladder-like manner–by using alternate down and forward motion. The height of the last step in the ladder determines the fraction of the T-shirt that is folded over. Similarly, one can define the *Lay-down backwards* and *Lay-down sideways* tasks, which only differ in the direction along which the cloth is laid on the table.

  In some cases, my system uses the table's edge for laying down the shirt. For example, to lay down a shirt backwards, the user moves the shirt beyond the forward edge of the table,

Figure 6.1: Critical points on the shirt for the shirt folding strategy. The locations of points A and B, along with the length and width of the shirt are the input to the controller. The controller infers the locations of points C and D at runtime.

lowers it to the appropriate height, and then moves the shirt back. (see [71]).

It is possible to define other more complex motion-tasks such as placing both arms simultaneously. However, the shirt-folding system does not require such motion tasks.

User tracking and feedback

I used a Vicon MX motion-capture system to track the user. I defined an 11-marker skeleton to track the user's arms and torso. Five markers in the skeleton defined the user's torso (four placed symmetrically on two shoulders and at the waist, and one placed asymmetrically to disambiguate between them) and the direction the user is facing. The torso markers defined the rotation of the body frame with respect to the global frame. Three markers were placed on each arm: one on the upper arm, one at the elbow, and one on the tracking finger. For each arm the marker on the respective shoulder acted as the origin of the local coordinate frame.

My system used 12 motors (six on each arm) to provide feedback. Every motor corresponded to one particular motion primitive (either in positive or negative direction along a canonical axis). The motors on one arm were connected to an XBee module [20]. The XBee modules for an arm and the motors shared a pack of three AAA batteries as their power source. Figure 6.2 shows the placement of the marker and motors on one arm for the shirt folding system. The feedback motors were run in the continuous feedback mode.

Folding sequence

The shirt-folding system folds a T-shirt using a sequence of folding steps. Each folding step involves performing one high-level motion task. The folding controller takes four parameters as inputs: the locations of points A and B, and the length (L) and width (W) of the shirt (figure 6.1). A and B are at the halfway point along the shirt's length. Folding a T-shirt involves the following sequence of motion tasks (refer to figure 6.1 for points and variables mentioned in the algorithm):

1. Place left hand at A.

2. Place right hand at B.

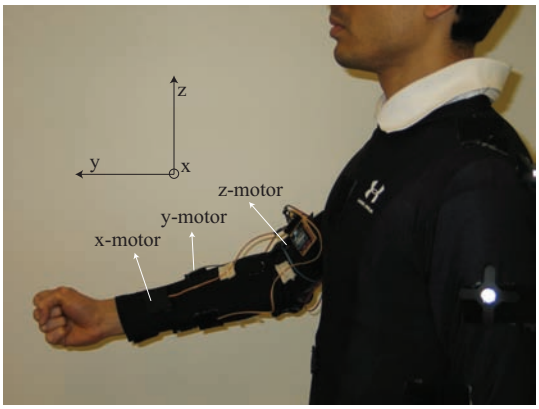


Figure 6.2: The body-coordinate frame for the shirt-folding system. The motors are placed at positions such that they "push" the arm in the intended direction.

3. Lift the shirt up to a height $L/2$ (Move parallel along Z task). This move halves the shirt along its length.
4. Lay the halved shirt on the table (Lay-down backwards). Calculate the position of points C and D from the current position of the left and right hand. C lies on the line connecting the two hand positions at a distance of $W/4$ from the left hand. D lies on a line perpendicular to that joining the two hands at a distance $L/2$ (see figure 6.1).
5. Place the left hand at C.
6. Place the right hand at D.
7. Lift the shirt up to a height $L/2$ (Move parallel along Z task).
8. Rotate the shirt by 90° in the clockwise direction. (A sequence of alternating place left and place right tasks.)
9. Lay the shirt backwards to a height $W/4$ (Lay-down backwards).
10. Lay the shirt forward on the table (Lay-down forward task).

6.2 Experiment and discussion

The user's skeleton was calibrated using the motion capture system. The dimensions of the shirt, its position and the height of the table were the inputs to the shirt folding system. Intuitively, the size and location of the T-shirt are the minimum information needed for any system that folds a T-shirt. For better folds, we used the table's edge for laying the T-shirt down, and so the position of the table's edge was also an input to the system.

The user was blindfolded and asked to follow the instructions from the shirt-folding system. Figure 6.3 shows a sequence of figures from one run of a user folding a T-shirt (The entire video for the folding sequence in figure 6.3 is available at [71]). I tested the shirt folding system for multiple runs at different times. Users could consistently fold the T-shirt using the system's feedback, and the average time for the fold was between two and three minutes.

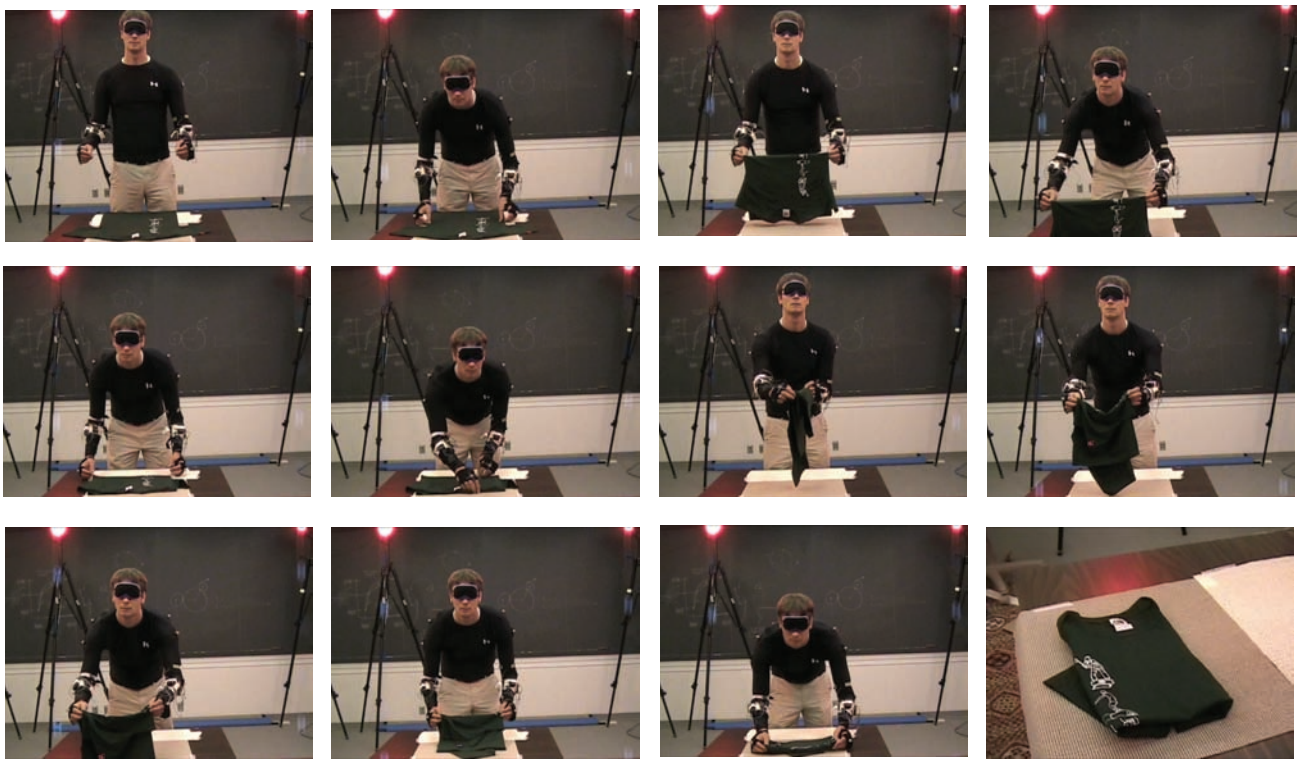


Figure 6.3: Different steps involved in folding a T-shirt. Our systems guides a blindfolded user through a sequence of manipulation moves that fold the T-shirt. Note: not all moves are shown in this figure.

The shirt folding application illustrates some important design philosophies of my systems. My systems use natural hand motions as control primitives. Therefore, no special training was required for the user to learn the primitives. This system extensively uses the human element in the control loop. The human arm is a complex, high-degree-of-freedom system, and moving it in any of the three control directions involves articulating multiple joints. The shirt folding system, however, does not solve the inverse-kinematics problem because it is "naturally solved" by the user. Similarly, grasping cloth is an extremely difficult task for robots that has only recently become somewhat feasible ([8], [37]), but humans have no problem in grasping and manipulating cloth.

The shirt-folding system can guide the user to place a motion-capture marker within one cm of the target location. This accuracy is extremely useful, *e.g.*, in ensuring that both hands lift the T-shirt to exactly the same height so the shirt is laid down symmetrically on the table.

The motion capture system was the source of most errors. The Vicon MX system frequently "lost" markers and is susceptible to bright light sources and reflective surfaces. The most serious problem was the mislabeling of markers. The shirt folding sequence sometimes caused the user's arms and fingers to cross each other. This crossing sometimes caused the system to mislabel the markers.

Although minimizing the time it takes to fold a T-shirt was not a goal, two to three minutes may seem a little long for folding a T-shirt. However, the folding sequence has ten steps, some of which are made of multiple motion primitives. Also, the shirt folding system waited for two seconds between two steps. Therefore, users change primitives fairly quickly in the folding sequence.

6.3 Lessons learned

Motion-capture systems provide accurate tracking of marker positions. However, marker labeling is a persistent problem with the Vicon MX systems. This problem is extremely difficult to solve, because the human body allows a wide range of motions. The marker labeling algorithm for motion-capture systems solve this problem by optimizing for multiple criteria such as fitting model parameters, adding motion data, recent position of markers, etc. I observed that motions where two markers came together and then diverged lead to many mislabeling errors. This behavior is not surprising, because when two markers are close their labels can be interchanged without significant changes in the error metric. Further, an erroneous labeling is propagated until the motion-capture skeleton's fit becomes poor. Losing one marker has a similar effect; the system optimizes the markers positions with insufficient data. As a result, I found it best to define folding steps that minimized such ambiguities for the labeling algorithm.

A related problem concerns the difference between the position of a marker and the point on the body the marker represents. For example, the marker representing a the tracked finger is still some distance from the fingertips that grab the cloth. This marker cannot be placed too close to the fingertip, because the cloth might occlude it. This gap combined with the flexibility of the human body introduces an unpredictable error in the system. Fortunately, human subjects intuitively understands the task and can partly compensate for this error.

The shirt folding system validates our assumptions about the human element in the control loop. The Cartesian controls were easily executed by the users without calculating the inverse kinematics for the arm. Similarly, grabbing flexible objects is not difficult for humans. I believe that leveraging human capabilities is an important lesson for future applications. Systems that take advantage of human abilities can achieve complex tasks with relatively simple tracking and feedback.

The hybrid control strategy of having a predefined sequence of tasks whose targets are defined at runtime worked well. This strategy defines the sequence of tasks and how their target configurations are related to each other. The actual planning for each task takes place at runtime. This is a general concept and may be extended to other applications. For example, a system that assists people in mixing ingredients for baking might specify the sequence in which the ingredients must be mixed. The planner could calculate the target configurations at runtime and guide the user accordingly.

Chapter 7

Posture Shirt

Many human motion tasks require people to maintain a fixed posture. For example, it is important for piano players to maintain good posture while playing (see Mora *et al.* [51]). People working at a desk for extended durations are often advised to maintain a *good* posture to avoid problems such as chronic back aches. In the absence of external feedback, many people slowly drift from a good posture to a bad one, without noticing the change. This chapter describes the design and implementation for a system that assists people in maintaining a specified torso posture.

The systems described in chapters 4, 5, and 6 were prototypes that explored the range of possibilities for motion-tasks using a motion-capture system to track the user. The posture shirt and the systems presented in chapters 8 and 9 address trajectory tasks. These systems use WiTilt [75] three-axis accelerometers to track the users. Compared to a motion-capture system, WiTilts are cheaper, more portable, and more reliable sensors. Consequently, systems that user WiTilts are more practically viable, and easier to deploy and test. Therefore, I have done more detailed user-testing for these systems.

7.1 Introduction

The posture system tracks the inclination of the user's torso. The inclination is measured along two axes shown in figure 7.1. These angles measure the amount by which the user bends forward or leans sideways. The posture system defines an ideal state and acceptable thresholds along all measured axes around the ideal state. If the user strays beyond these thresholds, he is prompted to correct his posture. Although users can customize the ideal torso position and thresholds, I fixed these values for the user experiments described in section 7.3.

The present system defines acceptable postures as a range of orientations for the user's torso[1]. My system estimates the torso's inclination by tracking the orientation of the user's upper back. The goals for the posture systems are as follows:

- **Maintain a posture**: The posture system should restrain the user's torso inclination to a small range around the ideal orientation.
- **Cheap, accurate and mobile system**: Build a portable system using cheap and accurate devices.
- **Minimally intrusive and minimally-distracting feedback**: The feedback should be sufficient to achieve the system's goals, but not too excessive to frustrate the users.

[1]This definition of good posture may not match the *clinical* definition of a good posture. It may be possible to develop systems that enforce clinically defined good postures.

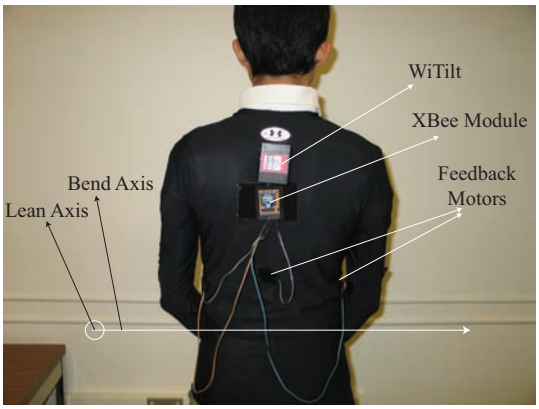


Figure 7.1: The tracking and feedback systems. The posture system measures the inclination around the bend and the lean axes. WiTilts track the user's orientation and the feedback motors provide appropriate feedback.

- **Permit small transient motions**: People shift, bend, or slightly move even while working at a desk. The feedback algorithm should not provide feedback for such transient motions.

The posture system uses accelerometers to track the user and Lilypad [13] vibration motors to provide feedback. These devices are mounted on a shirt. Thus, the system is a portable motion feedback system that uses off-the-shelf, inexpensive tracking and feedback devices. Section 7.2 describes the design of the system. I tested the posture system with ten volunteers. Sections 7.3 presents the results of these experiments. Section 7.4 evaluates the limitations of the current system, and identifies areas of future work. I will conclude in section 7.5 with lessons learned.

7.2 System design

In this section, I will briefly review the three major components for the posture system: the tracking devices and their data, the feedback system, and the controller algorithm.

Tracking system

The posture system tracks the inclination of the user's upper back. It measures the inclination of the back along two axes: one running along the waist, and the other parallel to the ground and perpendicular to the first axis. Figure 7.1 shows these two axes. I will refer to the rotation around the first axis (running along the waist) as the *bend* in the user's torso, and the inclination along the second axis as the *lean*.

The posture system uses WiTilt [75] wireless accelerometers (see chapter 3 for details about WiTilt) to track the orientation of the user's back. One WiTilt was placed on the user's upper back, just below the neck (see figure 7.1). In this orientation, the X and Y axes of the accelerometer were aligned with the user's back; as the user bends or leans sideways, the changes are measured by the accelerometer. My system defines bad configurations as deviations from the ideal positions along these two axes. It may be possible to assume *bad* postures without changing the upper back's inclination, *e.g.*, by twisting one's back. Such postures are acceptable configurations for the current system. However, it should be easy to extend the current system, probably with additional sensors such as inertia measurement units, to detect and correct more expansive definitions of unacceptable postures.

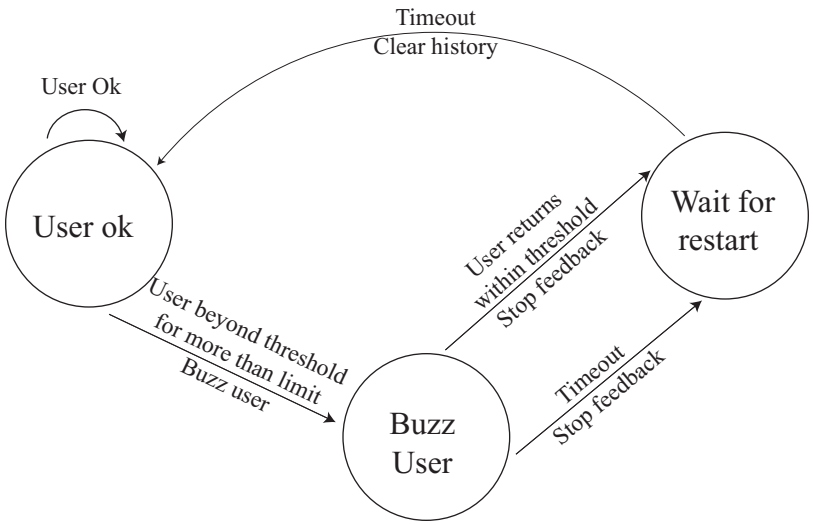


Figure 7.2: A state transition diagram for the posture system. The text above an arrow shows the transition condition, and the text below the arrow shows the action taken while transitioning. The controller uses time as a heuristic for deciding on feedback. If the user remains outside of the permitted threshold, the system buzzes the appropriate motor. After buzzing the controller enters a waiting state before restarting to track the user.

Feedback system

I used four Lilypad vibration motors [13] as feedback devices. These motors were placed on the abdomen, the lower back, and two sides as shown in figure 7.1. The controller buzzes a motor when the user's posture exceeds the threshold in that direction. For example, when the user leans too far to the left, the motor on the left side buzzes.

Since this system enforces corrective feedback, a motors only buzzes till the user's posture is outside the acceptable threshold. These motors are controlled using an XBee RF module [20] that is mounted on the user's clothes.

Controller design

The WiTilt mounted on the user's back conveys the user's torso orientation to the posture system's controller. Based on a history of the user's state, the controller decides what, if any, corrective feedback to provide. Collecting the user's state information is straightforward. Similarly, once the controller determines the necessary feedback, delivering it via the feedback and communication systems is simple. Most of the controller's design focuses on determining when to provide the user with corrective feedback.

My controller's design is based on three feedback philosophies: non-intrusiveness, accuracy, and forgiveness for transient motions. Non-intrusiveness implies that the feedback should be delivered to cause minimum distraction. Accuracy demands that the controller provide corrective feedback whenever the user assumes a posture that is not permitted. The guiding principle for forgiving transient motions is usability. A system that buzzes a motor every time a user is beyond a threshold configuration is not desirable. The errors in the user's posture could be due to temporary sensor noise. In such a case, the user would get incorrect feedback. Users frequently bend forward or lean sideways even while sitting, *e.g*, bending forward to pick a glass of water. If the user's configuration changes beyond the threshold during such motions, the controller should not provide

corrective feedback.

However, distinguishing transient motions from bad postures is difficult for systems with limited sensing. In some sense, forgiving transient motions enhances the usability of the system at the cost of accuracy. My system uses a simple time-based heuristic to separate bad postures from transient motions.

Figure 7.2 shows the finite-state-machine-based controller algorithm. The controller starts in the *User ok* state, and uses a history of the user's states for calculating the required feedback. If the user continuously strays beyond the threshold orientation for more than δ seconds (for a fixed value δ), the controller switches on the appropriate feedback motor and transitions to the *Buzz user* state. If the user returns to an acceptable posture or stays in the current (incorrect) posture for more than a timeout duration, the controller stops the feedback and transitions to the *Wait for restart* state. After waiting a fixed duration in the *Wait for restart* state, the controller clears its history and again transitions to the *User ok* state.

The controller allows the user to stray outside the threshold for δ seconds before providing feedback. This delay is useful in allowing short-term, transient motions such as picking up objects. Once it has buzzed the user for a fixed time, the controller stops and waits before starting the feedback loop again. This second delay in providing feedback allows users to finish actions that require a little more time such as tieing shoelaces. Overall, avoiding to constantly buzz the user increases the system's usability.

7.3 Experiments and results

I tested the posture system on ten volunteers. In this section, I will present the details of my experiments: the experimental setup, results measuring my system's performance, and the results of a user feedback survey I conducted.

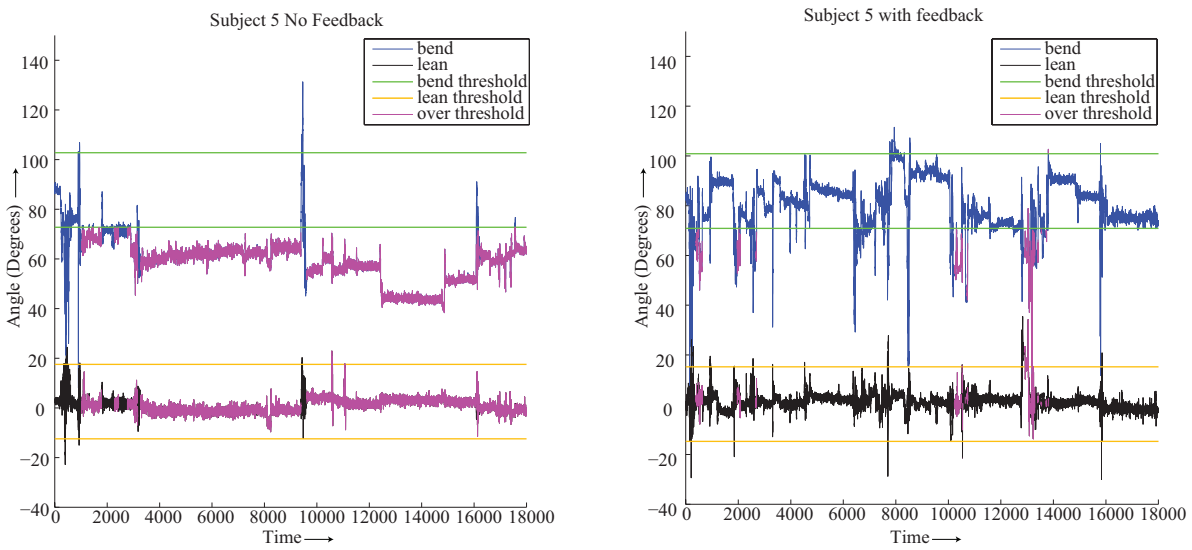
Experimental setup

I conducted the experiment on ten volunteers. Before starting the experiments, I explained the goals of the experiment and the details of the tracking and feedback devices to the users. The experiment consisted of two parts, each lasting ten minutes. In the first part, my system measured the user's torso's orientation without providing feedback. In the second part, in addition to tracking the user's orientation, the posture system provided the user feedback to confine them to the desired target orientation. I calibrated the users at the beginning of each part. The calibration process measured the reference accelerometer values for the ideal orientation; the user's current state was measured relative to this ideal value.

My experiments used a δ value of three seconds and a timeout duration of five seconds before restarting the feedback loop. I used a threshold value of 15° along all four directions; the user was allowed to bend 15° forwards, backwards and on either sides. All subjects were requested to sit at their desk and work normally during the experiment.

System results

Figure 7.3 shows the results of using my the posture for one user. The two green and yellow horizontal lines correspond to the bending and leaning thresholds for the user. Ideally, the user's bending and leaning values must remain between thse two horizontal lines. The blue and black curves in the graphs correspond to the user's bending and leaning angles measured by the sensor.



(a) User data without feedback

(b) User data with feedback

Figure 7.3: Example results for one subject using the posture system. (a) User's posture data without feedback. (b) User's posture data with feedback. The portions where the user strayed beyond the threshold is shown in pink. The green and yellow lines show the threshold values for the bending and leaning of the torso. Ideally, the user should stay between those lines at all times.

The pink portion of these curves corresponds to periods when the user was not in an acceptable orientation.

It is clear from figures 7.3(a) and figure 7.3(b) that the user strayed beyond the acceptable threshold significantly more without feedback. In figure 7.3(b), within a short time of straying over the threshold the user returned to an acceptable configuration. For all ten users, figure 7.4 shows the percentage of time the users were sitting in a bad posture both with and without feedback. The black curves in figure 7.4 (corresponding to the feedback case) is significantly below the blue curves (corresponding to the case with no feedback), with the difference between them sometimes close to 90%.

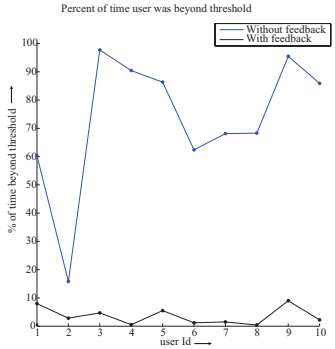
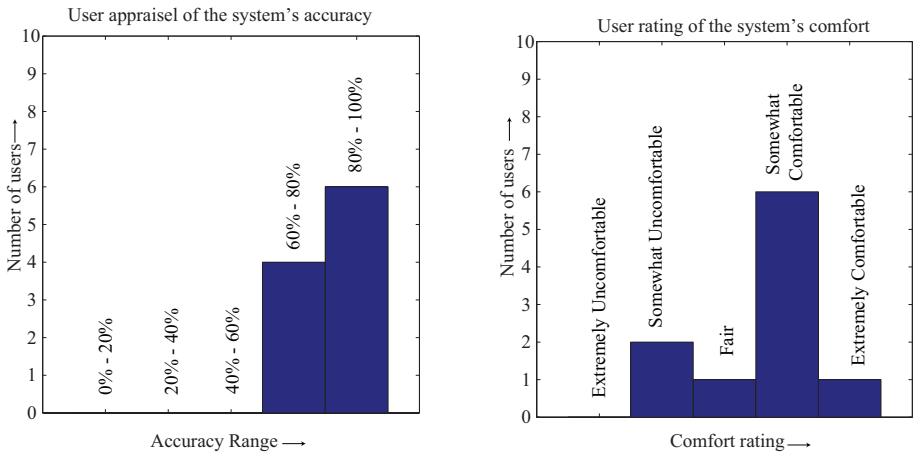
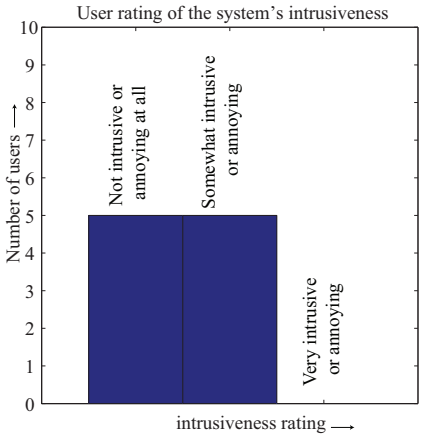


Figure 7.4: Aggregate results for all ten users. The blue curve shows the percentage of time a user strayed beyond the permitted threshold without any feedback. The black curve shows the percentage of time the user strayed beyond the threshold using the feedback system.



(a) User feedback results on accuracy

(b) User feedback results on comfort



(c) User feedback results on intrusiveness

Figure 7.5: Results to questions from a feedback survey administered to the users. (a)How accurate was the feedback? (b)How comfortable was the system to use? (c) How intrusive or annoying was the feedback?

User-experience results

After testing my system, I asked the users to answer a questionnaire about the system's usability and effectiveness. Figure 7.5 shows the answers to some questions in the feedback survey. Six of the ten users rated the system's feedback as highly accurate (80%-100%) and the others rated it as accurate (60%-100%) (figure 7.5(a)) . Similarly, seven of the ten users rated the systems as comfortable or somewhat comfortable (figure 7.5(b)). Figure 7.5(c) shows that half the users found the vibration feedback *somewhat intrusive or annoying*, while the other half did not find it intrusive at all. Eight of the ten users said they would be somewhat likely or extremely likely to use this system for an extended duration (results not shown). All ten users agreed that the system was effective in helping them maintain the desired posture (results not shown).

I sought the user's comments on different aspects of the system. Many users liked the simplicity, accuracy, and non-intrusiveness of the system. Three users also liked the delayed-buzzing algorithm. The most unpopular aspect of the system were the tight shirts. In the suggested improvements section, some users desired a way to attach the motors on top of normal clothing.

7.4 Limitations and future work

Although the results for the system's efficiency and user experience are encouraging, my experiments are a small-scale study. Future studies with larger sizes and longer durations would be needed to strengthen the current results.

Another possible extension might be to extend the current system to enforce a clinical definition of good posture. My system provides a good starting point for such applications. The current study was restricted to users working at a desk. Activities such as standing and walking also require good posture. It should be possible to extend the current system to newer activities.

Instead of fixing parameters, such as the bending threshold, future systems might learn these parameters. For example, such a system may seek the user's response every time it provided feedback, and learn the user's preferences over time. However, repeatedly asking user's feedback would make a system more intrusive and may deter users from using the system for long durations.

The current system needs a computer to run the controller program and interact with the sensors. However, microcontroller boards such as Arduino BT [3] can perform the same function. Future systems may be entirely self contained: the user merely wears the shirt that has the sensors, feedback devices and a microcontroller board all mounted on it.

7.5 Lessons learned

My system allows users to customize different parameters such as thresholds for orientation tolerance, and delays for feedback. However, to ensure uniformity across subjects, I had to fix these values during user testing. Selecting such uniform values is difficult. For example, some users praised the delay in giving feedback, but one user thought the duration was too small. Fixing usability parameters can skew the user feedback results.

Although the feedback is viewed favorably, some users were unhappy with the shirt because it was too tight and could not be worn on top of regular clothing. A different and more usable shirt design would have made the system more user friendly. The posture system needs to balance accuracy versus usability: tighter shirts allow more accurate tracking, but are uncomfortable to wear

Chapter 8

Arm Motion Controller

The arm-motion feedback system assists users in precisely performing a particular arm motion. This system provides corrective feedback for trajectory tasks, as opposed to the restricted feedback for configuration-task systems such as the shirt folding system described in chapter 6. Trajectory tasks assume that the user understands the motion, but needs assistance in performing the motion precisely.

Many motions have the following structure: perform an action to get to a desired configuration, hold that pose for a fixed amount of time, and move on to another (or the same) action. Consider the action of squeezing a rubber ball for rehabilitation. The user squeezes the rubber ball till a target configuration, then holds the ball in that states, and finally releases it. Achieving the precise pose and holding it for a precise time may be deceptively difficult. Furthermore, the motion description may require the user to move only certain body parts while maintaining other body parts in a fixed configuration. The arm-motion feedback system addresses all three issues mentioned above:

- **Precision**: The system guides the user to a precisely defined target pose.
- **Timing**: Once the user has achieved the desired pose, the system provides feedback to maintain that pose for a specific duration.
- **Form**: The controller provides feedback that constrains the non-active body segments to a fixed configuration.

The rest of this chapter describes the design, implementation, and experiments for the arm-motion feedback system. Section 8.1 presents the design of the three major components of the system: tracking system, controller, and feedback system. Section 8.2 describes experiments and results for the arm-motion feedback system. Section 8.3 discusses some limitations of the current system along with possible extensions. I will conclude in section 8.4 with the lessons learned while developing this system.

8.1 System overview

The arm-motion feedback system consists of three parts: a tracking system, a controller that calculates and conveys the appropriate feedback based on the user's current state, and a feedback system that provides the corrective feedback. In this section, I will briefly describe each of these components.

8.1.1 Controller design

The human body consists of multiple degrees of freedom. For a given action, these degrees of freedom maybe divided in two classes: active and constrained. Active degrees of freedom change their state over the course of the motion, while the constrained degrees of freedom maintain their state. Usually, a motion segment terminates when the active degrees of freedom achieve their target configuration. For some motions, it may be necessary to maintain the constrained degrees of freedom in a fixed configuration. Once the user reaches the target configuration, he may maintain that pose for a fixed duration before continuing on to the next motion segment. Calisthenics provide good examples of such a motion paradigm. For example, in a squat, the knees are the active degree of freedom while the waist (the legs-torso joint) is the constrained degree of freedom. The user bends his knee joints to the target angle, while keeping the torso upright. The user maintains this target configuration for some fixed duration, before straightening the knees and going back to the original position.

Task specification

The arm-motion feedback system treats the expected motion as a sequence of motion segments. Every motion segment has at least two *tasks* associated with it: a foreground task that contains the details for the active degree of freedom, and a set of background tasks for the constrained degrees of freedom.

Every task has two attributes: a terminating condition and a waiting time. For foreground tasks the terminating condition specifies the target configuration for the active degrees of freedom. The waiting time specifies the time that the user must remain in the target configuration before moving to the next motion segment. For background tasks, the terminating condition specifies the configuration that the constrained degrees of freedom must maintain for the duration of the motion segment. The waiting time for background tasks is zero.

The user defines a motion by specifying the sequence of motion segments and the foreground and background tasks associated with each segment. The controller processes the motion segments sequentially, and provides corrective feedback.

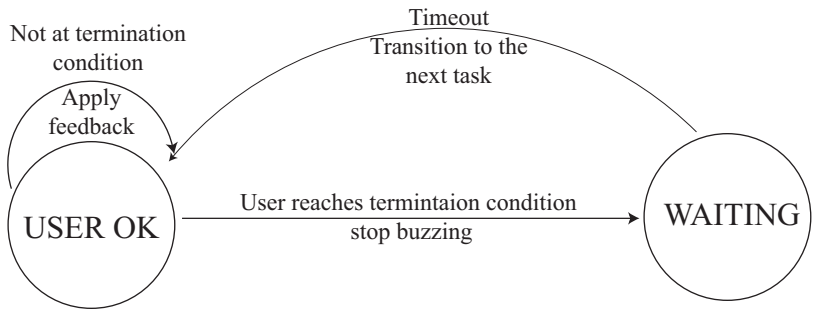
For both foreground and background tasks, the controller allows an error threshold. The arm-motion feedback system guides the user's pose to within this error threshold of the target configuration.

Controller algorithm

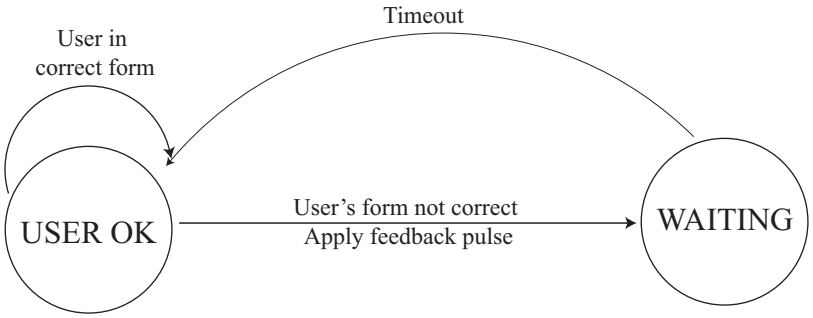
The controller calculates the feedback for foreground and background tasks separately. It models the algorithm for calculating the feedbacks as a finite-state-machine (FSM), where the output for each state depends on its current state and the sensor values[1]. Figures 8.1(a) and 8.1(b) show FSMs for calculating the foreground and background task feedback respectively.

The FSM that calculates the feedback for foreground tasks has two states: a USER OK state and the WAITING state. In the USER OK state, if the user's current state meets the target configuration the controller immediately transitions to the WAITING state. Otherwise, the controller provides appropriate feedback until the user reaches the target configuration. In the WAITING state, the controller waits for the time specified by the foreground task. After waiting for the required time, the controller prompts the user to transition to the next motion segment

[1]Our algorithm most closely resembles a Mealy machine, where sensor values are the input to the current state and the feedback motor to buzz is the output value.



(a) Finite state machine for foreground tasks



(b) Finite state machine for background tasks

Figure 8.1: Finite state machines for the foreground and background tasks. The text above the transition arrows specifies the condition that lead to the transition. The text below the arrow specifies the action that the controller takes while performing that transition.

The FSM for calculating the background feedback is different from that for foreground tasks. Once the controller determines that a background tasks requires feedback, it provides the appropriate feedback and immediately transitions to the WAITING state. The controller remains in the WAITING state for a fixed duration, before transitioning to the USER OK state. Unlike foreground tasks, the timeout in the WAITING state for background tasks is a constant.

Figure 8.2 shows a flowchart of the controller's overall algorithm, where figures 8.1(a) and 8.1(b) show the algorithms used to determine the feedback for the foreground and background tasks respectively.

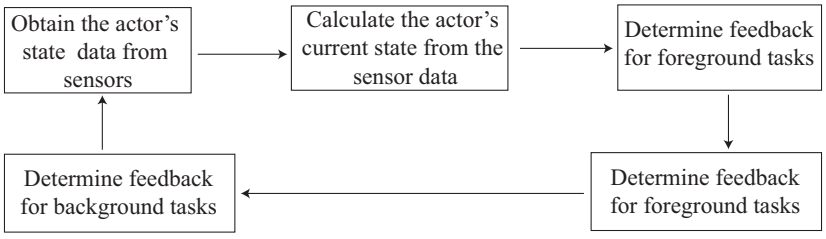


Figure 8.2: A flowchart showing the major components of the controller algorithm. Figures 8.1(a) and 8.1(b) show the algorithms used to determine the feedback for the foreground and background tasks respectively.

8.1.2 Tracking devices

I used WiTilts [75] to track the user's arm configuration. The WiTilts were mounted on the user's upper arm and forearm to track the two segments. Section 8.2 presents the configuration space variables for my application along with the details of calculating these variables from WiTilt data.

Data filtering

WiTilts provide accurate acceleration data about their three axes, but this data is noisy. This noise causes spikes in the acceleration values, even when the underlying motion is smooth. The controller mitigates the effects of noise by using a low-pass filter. A low-pass filter calculates a weighted average of the sensor data in a history window. The controller provides feedback based on this average instead of the actual sensor values.

Let x_i represents the ith sensor value in the sensor's data stream. Also, let y_i be the value the controller uses to determine the feedback. The value y_i is obtained by processing x_i. The controller assumes that the sensor data should not change substantially between two successive readings. This assumption is valid if the sensor is polled at a sufficiently high frequency and the underlying motion is not too fast. Both of these assumptions are valid for the current system. Under these assumptions, y_i is obtained from the raw sensor data, x_i, as follows:

$$y_i = \begin{cases} (1-\beta(i))y_{i-1} + \beta(i)x_i & i > 1, \\ x_i & i = 1, \end{cases} \tag{8.1}$$

for $0 \leq \beta(i) \leq 1$. The new filtered value, y_i, is a weighted average of the older processed value (y_{i-1}) and the new sensor value (x_i), where the relative weights of the two terms are determined by $\beta(i)$. The magnitude of $\beta(i)$ measures the confidence in the new sensor reading (x_i). In the extreme cases, $\beta(i) = 1$ implies complete confidence in the new sensor readings, whereas $\beta(i) = 0$ implies no confidence. $\beta(i)$ is calculated using the following formula:

$$\beta(i) = \max(\beta_{\min}, 1 - \delta x_i/\delta_{\max}) \ ,$$

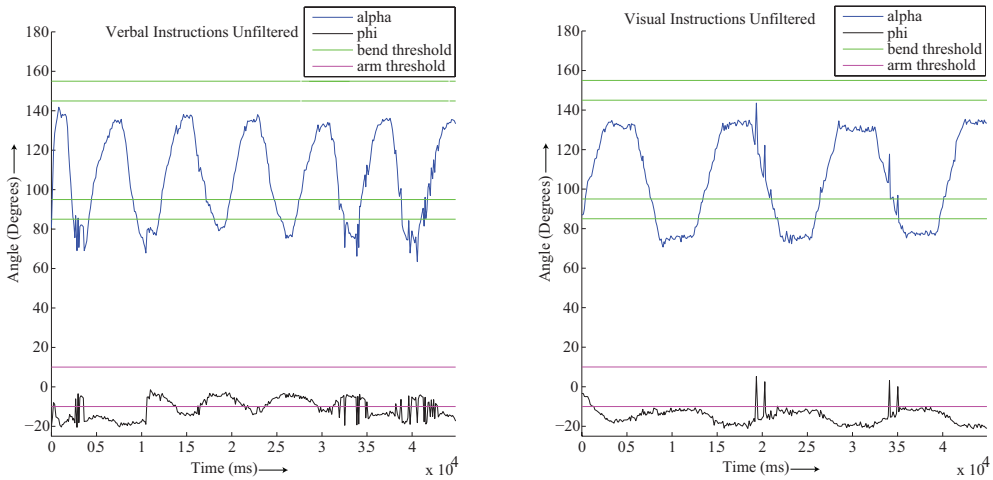
where δx_i is the absolute value of the change in the sensor reading given by

$$\delta x_i = \min(\delta_{\max}, |y_{i-1} - x_i|) \ ,$$

$\delta_{\max}$ is the maximum permissible change in the sensor reading, and $\beta_{\min}$ is a lower bound on the weight of the new sensor reading.

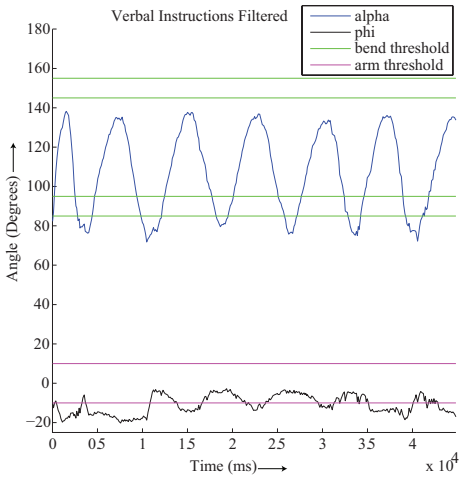
This formula is based on the following principle: the more the value of x_i differs from y_{i-1}, the lower the confidence in its accuracy. Consequently, as the difference between y_{i-1} and x_i increases, the value of $\beta(i)$ increases. Also, the bounds on $\beta(i)$ ensure a minimum weight for the new sensor value.

Figure 8.3 shows the effect of filtering on the sensor data. The α and φ values represent the bend at the elbow and the elevation of the upper arm respectively. Figures 8.3(a) and 8.3(b) show the raw sensor values for the two angles. This data contains minor spikes due to noise. Figures 8.3(c) and 8.3(d) show the same data filtered using equation 8.1. The filtered data is smoother and follows the general data trend better than the raw data.

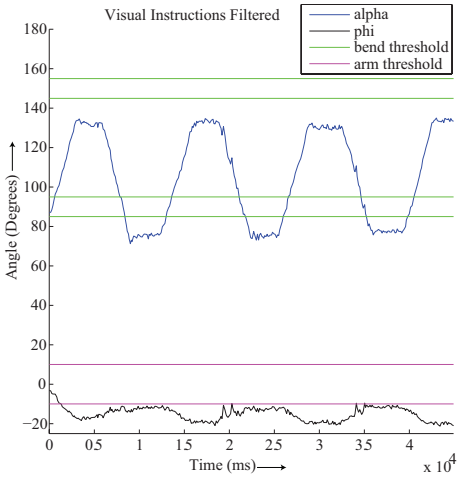


(a) Unfiltered data example 1

(b) Unfiltered data example 2



(c) Filtered data example 1



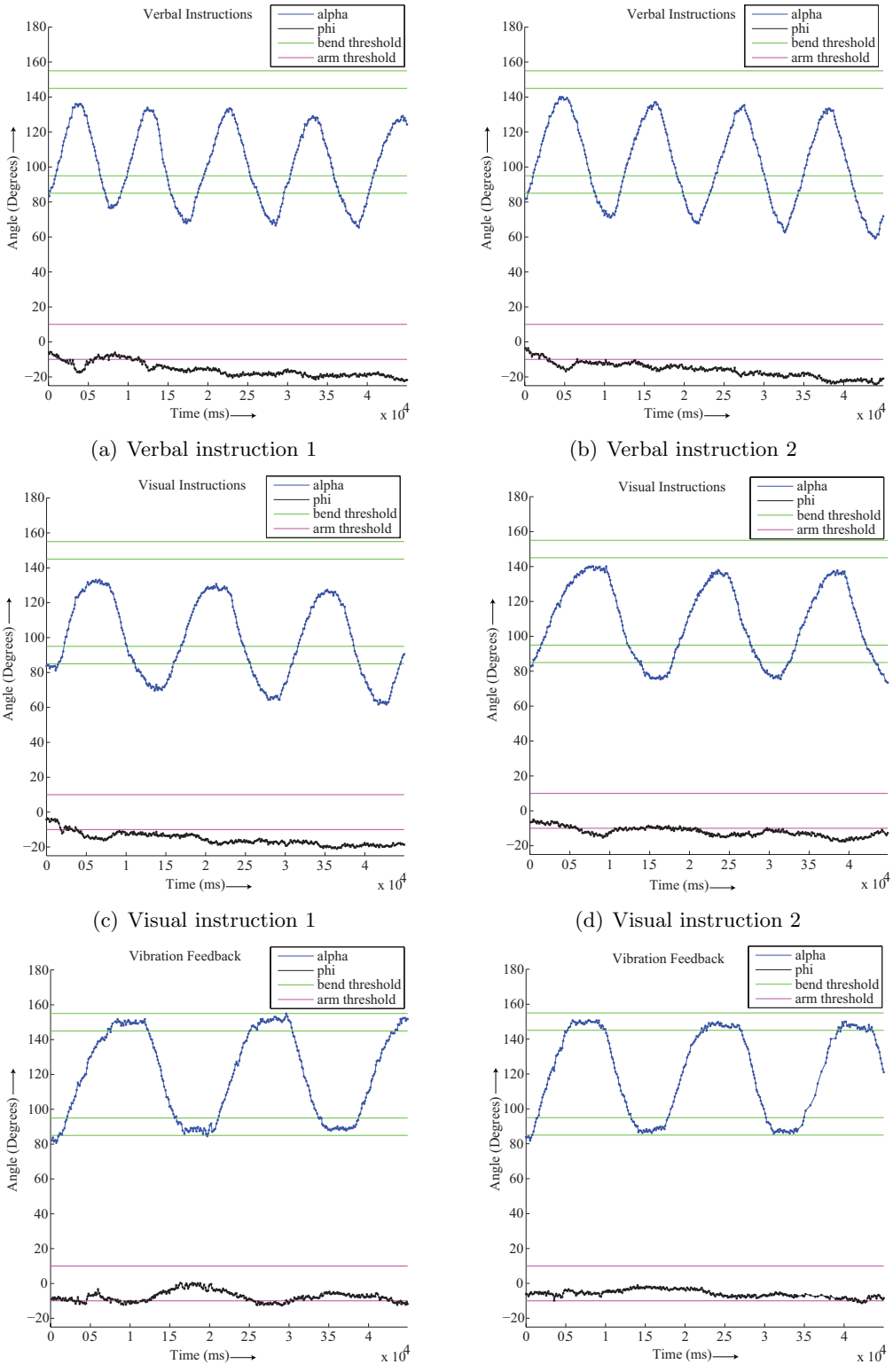
(d) Filtered data example 2

Figure 8.3: Examples of filtering the data using the low-pass filter. (a) and (b) represent the raw arm bend and elevation values (α and φ respectively). (c) and (d) show the corresponding filtered values for the data in (a) and (b).

8.1.3 Feedback devices

The arm-motion-feedback system uses Lilypad vibration motors [13] for providing motion feedback. The Lilypad motors were controlled using XBee radios (refer to chapter 3), which were also mounted on the user's body. The vibration motors were used in two modes: continuous mode for the active degrees of freedom and pulsing mode for the constrained degrees of freedom.

Pulsing mode requires the feedback software to run asynchronously with the tracking and control software(refer to chapter 3). The arm-motion feedback system solves the asynchrony problem with a client-server architecture. The feedback system starts a server that listens to commands from the controller, and controls the feedback motors accordingly. A client-server architecture offers several advantages. The details of communicating with the XBees are abstracted from the controller. For large systems, the client and server may run on separate machines, freeing the client system's resource for tracking and controller software.



(a) Verbal instruction 1

(b) Verbal instruction 2

(c) Visual instruction 1

(d) Visual instruction 2

(e) Vibration Motor Feedback 1

(f) Vibration Motor Feedback 2

Figure 8.4: Examples of the arm motion under different feedback conditions. (a) and (b) No feedback, only verbal description. (b) Visual instructions from watching an example video. (c) Vibration feedback with my system. All figures are from different trials of the same subject. α refers to the bend in the elbow and φ refers to the elevation of the upper arm. The green horizontal lines show the tolerated threshold for the error in the elbow bend, and the magenta lines show the tolerated threshold for the upper arm elevation.

Figure 8.5: WiTilts mounted on the user's hand to measure the arm's configuration.

8.2 Experiments and results

I tested my system with human users for a particular arm motion. Section 8.2.1 describes the details of the experiment: the motion that users were tested on and the protocol used to test the system. Section 8.2.2 presents the results of the user experiments, and section 8.2.3 describes the results of a user feedback survey conducted as part of this experiment.

8.2.1 Experiment design

I tested the arm-motion-feedback system for a simple repetitive arm motion. The motion involves moving the forearm to bend/extend the elbow joint. This motion is similar to an arm-curl in weight lifting. The motion involved performing two motion segments sequentially. The foreground task for the experiment was the motion of the forearm to control the bend in the elbow joint. The first motion segment involved extending the forearm so that the bend in the elbow joint was 150°. The second motion segment involved bending the elbow joint until the bend was 90°[2]. These angles are *natural* angles, and so they are easy for the users to understand. The waiting time between the two tasks was three seconds.

For each of the foreground tasks, the associated background task was to maintain the upper arm parallel to the ground. This background task prevents the upper arm from *dropping* while performing arm exercises.

The complete specification for the task could be stated as: bend/extend the arm till the target configuration, hold the pose for three seconds and maintain the upper arm in a horizontal position all through the motion. I evaluated my system on all three aspects of the motion: accuracy of the foreground task, maintaining form for the background task, and holding the pose for the correct duration between tasks.

The configuration space for the controller is the angle of the elbow joint and the elevation of the upper arm. My system uses WiTilt three-axes accelerometers [75] to track the user's arm configuration. Two WiTilts, one mounted on the user's upper arm and another mounted on the user's forearm, were used to calculate the user's current state. These devices were placed such that their XY plane aligned with the plane of the user's upper arm and forearm (see figure 8.5).

WiTilts provide the elevation (pitch) and roll angles for the segments they are placed on. The elevation of the upper arm was the configuration space variable controlled by the background task.

[2]A fully extended forearm makes an angle of 180° and a fully bent forearm makes an angle of 0°.

The controller calculated the bend in the elbow using the elevation value of both the upper arm and the forearm. Choose a coordinate frame so that both the upper arm and the forearm lie in the XY plane. An arm segment represents a vector in the XY plane. Given the two vectors, representing the upper arm and the forearm, it is straightforward to calculate the angle between them.

The feedback system consists of four motors that control the foreground and background tasks. The motors for foreground task are place at the wrist, while those for the background task are placed on the upper arm.

8.2.2 System results

I tested the arm-motion feedback system on 10 volunteers. Every volunteer was asked to perform the motion under three feedback conditions:

- **Verbal instructions**: I verbally described the motion –the foreground, background tasks and waiting times– to the volunteer. The volunteer performed the motion based on the verbal description. This case is similar to reading an exercise description and performing it.

- **Visual instructions**: The volunteer was shown a video of the expected motion. From the video, the volunteers could estimate the amount of bend in the elbow, the positioning of the upper arm, and the waiting time. The volunteers were asked to perform the specified motion after watching the video. The visual instruction case is similar to watching a motion video and then trying to duplicate the motion offline.

- **Vibration feedback**: The volunteer was provided feedback with vibration motors as he performed the motion. This case tests the effectiveness of our system in providing corrective feedback for the motion task.

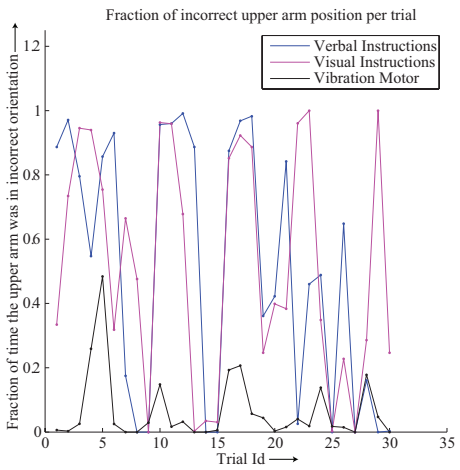
For each of the three feedback cases, each user was asked to perform the motion three times. Each of these trials lasted 45 seconds. Before every trial, I calibrated the user. During calibration the user was asked to hold the upper arm and forearm in a particular pose. The changes in the user's configuration were measured with respect to this reference configuration.

Figure 8.4 shows the results from one user for two trials each with verbal instructions (figures 8.4(a) and 8.4(b)), with visual instructions (figure 8.4(c) and 8.4(d)), and with vibration feedback (figures 8.4(e) and 8.4(f)). In these figures, α refers to the bend in the elbow joint and φ refers to the elevation of the upper arm. The yellow and magenta horizontal lines represent the error threshold around the target configurations for the foreground and background tasks respectively.

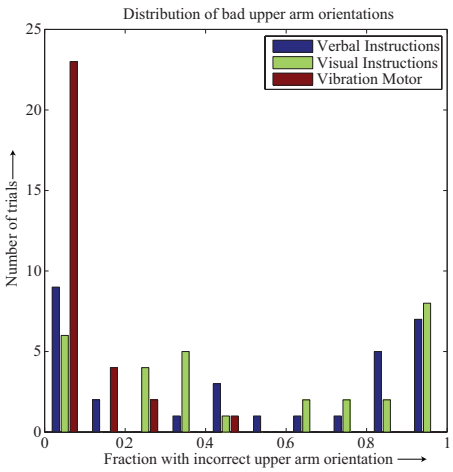
For an ideal motion, the upper arm should stay perfectly horizontal, and the φ curve should always remain between the error threshold lines. The α value should change monotonically till it reaches the target configuration (somewhere between the yellow lines). Once at the target configuration, this curve should remain almost flat for the duration of the waiting period. After the waiting period, the α curve should change directions to move to the target configuration for the next task. All six curves in figure 8.4 show periodic behavior, but they differ in accuracy.

For the examples shown in figures 8.4(a) and 8.4(b), the user does not bend the elbow by the correct amount, and the waiting period after each task is extremely short. After an initial period, the upper arm drops, as evidenced by the φ values dropping below the acceptable threshold.

Visual instructions improved the user's performance as shown in figures 8.4(c) and 8.4(d). The waiting interval between tasks is longer than in the case with verbal instructions, and the upper arm is closer to the required configuration (the φ curve is closer to the threshold). However, both the α and φ values do not meet their target configuration accurately. Interestingly, for both



(a) Fraction of time the background constraint was violated for every trial.



(b) Distribution of the fraction of time the background constraint was violated.

Figure 8.6: Results for the cumulative data for the background constraint. (a) The fraction of time the background constraint was violated for every trial run for the three different modes of feedback. (b) A histogram of the values shown in (a).

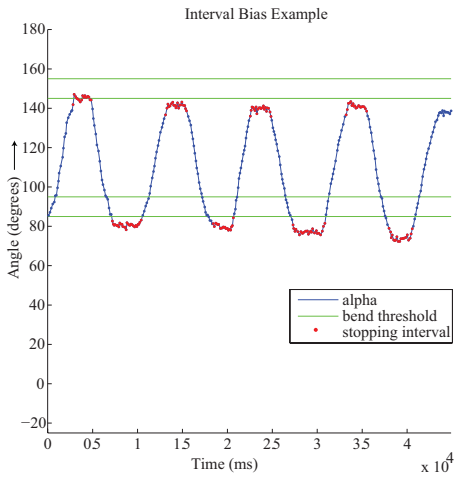
verbal instruction and visual instruction cases, the target α values for the foreground task are not consistent across different iterations.

Figures 8.4(e) and 8.4(f) show the user's performance when provided with vibration feedback. The user performs the task accurately. For the foreground tasks, the user's target configurations lie between the yellow line. The upper arm's elevation mostly remains within the threshold for the background task. The waiting interval between the successive tasks is consistent and closer to the desired value (three seconds) than for other modes of feedback.

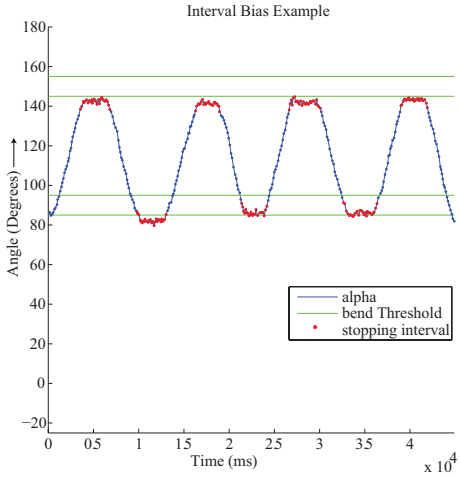
The examples shown in figure 8.4 are representative of the general performance for the three experimental conditions. Figure 8.6 shows the aggregate results for the violation of the background constraints. For every trial, it is easy to calculate the fraction of observations where the background constraint was violated. Figure 8.6(a) plots this fraction for all the user trials for the three modes of feedback. Figure 8.6(b) plots the data in figure 8.6(a) as a bar graph. For vibration feedback, most trials have a small (less than 10%) fraction of observations where the background constraint was violated. However, this bias is not shown by the other two modes of feedback. Both visual instruction and verbal instructions systems show an almost uniform distribution for the fraction of observations that violated the background constraint. These results show that real-time feedback was effective in assisting users to observe the background constraint.

Automatically measuring the terminating value and waiting period from the user data is extremely difficult. Although my system identifies the user's current state and task for providing feedback, this data cannot be used to measure the accuracy of the system. I designed a heuristic method to identify the terminating condition and waiting-interval length from the foreground task data. My method plots a best fit line between every point and successive data points that are at most 4000 ms away. If the line is almost horizontal (with slope below a small threshold), it implies that the user's state did not change significantly during that interval. This line is, therefore, retained as a line representing a potential terminating condition for a foreground task.

If starting at point i, multiple such lines can be drawn between i and $i+k$ (for different values of k), the longest such interval is chosen. Finally, if multiple intervals (starting ar different values of i) overlap, the longest amongst them is chosen. The duration of such interval gives an estimate of

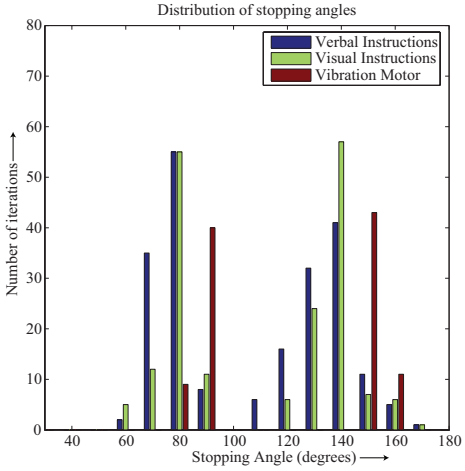


(a) Example 1 of interval chosen by the heuristic algorithm.

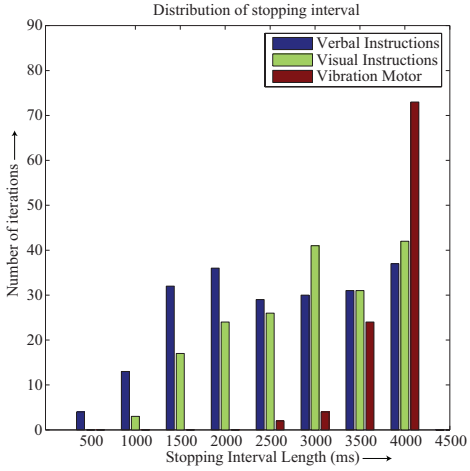


(b) Example 2 of interval chosen by the heuristic algorithm.

Figure 8.7: Two examples of the terminating intervals estimated by my algorithm. The red circles show the points that my algorithm chose as forming the terminating intervals for the user data.



(a) Histograms showing the termination conditions for the foreground tasks for the three modes of feedback.



(b) Histograms showing the length of the waiting periods between foreground tasks for the three modes of feedback.

Figure 8.8: Histograms showing the termination condition for foreground tasks, and the waiting interval between these tasks. The waiting time duration and termination condition were calculated using my heuristic algorithm.

the time the user spent in a fixed pose before transitioning. The average value of the points in the interval represents the target configuration achieved by the user. It is, thus, possible to estimate both the terminating condition as well as the waiting period for different foreground tasks from the intervals identified by my algorithm.

This method of selecting termination intervals clearly favors larger intervals over smaller ones. This bias towards larger interval is demonstrated in figure 8.7. The red circles in the figure show the termination intervals selected by my algorithm. Visually inspecting these intervals shows that the algorithm correctly identifies the approximate location of these termination intervals. A closer look reveals that the algorithm overestimates the length of the interval. It includes points that do not necessarily correspond to the user holding his pose at the end of a foreground task.

However, this algorithm provides a method to estimate the distribution of target configurations and waiting durations between foreground tasks. I calculated the termination condition and waiting time for all the intervals returned by the algorithm.

Figure 8.8 shows the results from the heuristic algorithm. Both verbal- and visual-instructions systems show a large spread in the termination angle and the waiting intervals between foreground tasks. On the other hand, the terminating angle for vibration feedback shows two sharp peaks at 150°and 90°, the terminating conditions defined by our task descriptions. Also, the waiting time between foreground tasks show a sharp peak around 4000 ms and rapidly decrease; users showed small variance in their waiting time when they were provided vibration feedback. This implies that my system prompts users to wait for a consistent duration. The wait time of 4000 ms is, however, slightly longer that the 3000 ms specified by the tasks. The greedy nature of my heuristic algorithm, which tries to find the longest waiting interval, may be partly responsible for the longer waiting intervals.

In summary, the arm-motion feedback system provided effective feedback for all three aspects of the motion: accuracy, timing and form. This system was able to elicit uniform performance across users, as shown by sharp peaks in the distributions for the three motion metrics.

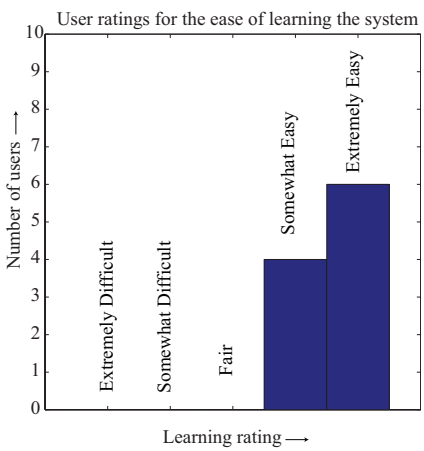
8.2.3 User-experience results

Figure 8.9 shows the results of user response to some feedback questions. All ten users responded that it was easy to learn the system (figure 8.9(a)), with six replying that it was extremely easy to learn the system. To the question "How comfortable was the system to use", nine of the ten users replied either *somewhat comfortable* or *extremely comfortable* (figure 8.9(b)). Further, eight users thought that the feedback was perfect for the application (figure 8.9(c)), whereas the other two thought it was somewhat lacking. All users thought that the system was helpful in improving their performance for the given task (results not shown).

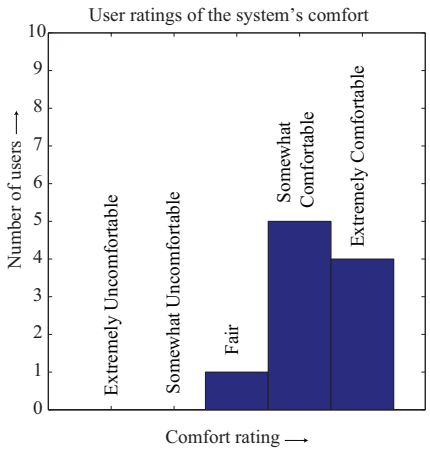
In addition to the above questions, I also asked users about the aspects of the system they liked or disliked. Most users liked the simple, accurate and realtime feedback and some felt it helped them identify hard to find errors. Two users responded that the system needs better delivery mechanisms such as smaller sensors and different shirts. Sports/gym applications were the most common response to a question that asked users about potential applications for this system.

8.3 System limitations and future work

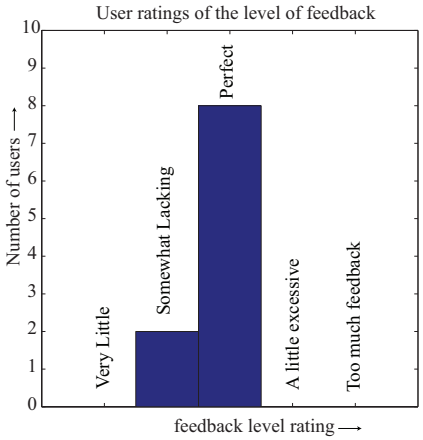
The results presented in sections 8.2.2 and 8.2.3 are encouraging, but the current system has some limitations. The arm-motion feedback system does not control the speed at which the user performs the motion task. Some motion tasks need to be performed not only precisely, but with a specific



(a) User feedback on the ease of learning the feedback



(b) User feedback results on comfort



(c) User feedback results on the level of feedback

Figure 8.9: Results to questions from a feedback survey administered to the users. (a)How difficult was it to learn the system? (b)How comfortable was the system to use? (c) What was the user's opinion on the level of feedback?

speed. The current system cannot be used for such motion tasks. Chapter 9 describes a system that provides users with basic velocity feedback.

Currently, the system ignores the user's reaction time when considering how long to wait before switching to the next task. This may lengthen the waiting period between foreground tasks, and may also contribute to longer waiting intervals observed in figure 8.8(b)[3]. A predictive system that takes user's reaction time into consideration may improve the system's performance.

I have tried the current system for a simple motion task. Trying this system for more complex motion tasks, possible involving other body segments such as the torso, is a promising area for future work. Calisthenics and other exercises provide excellent potential applications for such systems.

Future systems could integrate all components of the current system— tracking, controller and feedback–into one system. One way to achieve this goal could be to use a microcontroller board to perform all these functions. The microcontroller board could read data from the sensors, run the

[3]This is speculative, because any delay in responding to the *start* feedback should be canceled by the delay in stopping with response to the stopping of the motor's buzzing

controller algorithm to determine the corrective feedback, and control the vibration motors using its output pins.

8.4 Lessons learned

It is difficult to measure the user performance, especially the waiting interval between foreground tasks, from the user's performance data. I tried different heuristics to measure the waiting periods, but they all showed systematic bias. Hough Transform, a well-known method for identifying lines in an image, was not applicable in this case, because the data is sparse. I could demonstrate my system's consistency by showing that the distribution of waiting interval lengths (under systematic bias) has sharp peaks. However, this is a weaker result than showing that the waiting interval lengths match those specified by the task.

In absence of filtering, the sensor data frequently contains spikes due to noise (as shown in figures 8.3(a) and 8.3(b)). Such spikes can incorrectly trigger the background task controller to provide a vibration pulse. Filtering the data removes most such spikes, but does not affect the quality of feedback.

My system needs to be calibrated before every trial. Such frequent calibration may detract users from testing the system for extended periods. One reason for this repeated calibration is the small number of sensors that I use. Using multiple sensors and combining their relative data might provide an easier way to calibrate the user. However, more sensors need more space on the user's body and increase the complexity of the tracking algorithm.

Apart from the official feedback forms, informal interactions with the users revealed their preferences for different aspects of the system. Some aspects of the system such as buzzing motors (for foreground tasks) till the user reached the target configuration were universally liked. User opinion was divided over other aspects such as providing vibration pulses for background tasks even while the foreground motors were buzzing. Ideally, user feedback should be incorporated right from the development stage. Such a practice would lead to higher user satisfaction with the final system.

Chapter 9

Motion Feedback System to Synchronize Two Users

Human beings need to synchronize actions for many applications: rowing as part of a crew, dancing as a troupe, synchronized swimming, tug-of-war, etc. Individual actors need to synchronize their actions to properly execute these motions. Systems that synchronize users have the potential to greatly improve the performance of such motions.

Actions need synchronizing in many different contexts, each with their own challenges and objectives. For example, in rowing and tug-of-war all actors need to apply force at the same time. On the other hand, dancers in a troupe need to execute their actions in the proper sequence with precise timing.

I have built a motion-synchronization system that synchronizes two users. My system concentrates on one particular application: synchronizing the motion of two actors where one acts as a leader and the other as a follower. In this system, the actors cannot see each other, and the feedback from vibration motors is the only external cue to synchronize their motion. Further, only the follower receives feedback to match the leader; the leader is free to perform the action at his own pace. Practical examples of a "leader-follower" scenario are aerobics or dancing classes where students try to emulate their teacher's actions. In many instances, the followers may not see the leader, *e.g.*, in a dance lesson where they are facing away from the instructor, or yoga lessons where the yoga pose prohibits the students from looking at the instructor.

The rest of this chapter describes the design, implementation, and experiments for the motion-synchronization system. Section 9.1 describes the design of the major components of the synchronization system. Section 9.1 presents experimental results for a system that I implemented. Section 9.3 discusses the limitations of the present system, and future extension. I will conclude in section 9.4 with some lessons learned from my experience in building this system.

9.1 System overview

Before presenting the overview of the different components of my system, I will define some terminology. The motion-synchronization system has two users: a leader and a follower. In the rest of this chapter, the term *user* refers to either the leader or the follower. Human motions often consist of a sequence of motion segments, motion fragments with almost-constant joint-velocities. While performing a motion sequence, at any instant a user is performing one particular motion segment. I will refer to a user's current motion segment as the *motion state* of the user. The *motion state* for a user is different from the user's *state*, which refers to the current configuration (joint angles)

of the user.

The motion-synchronization system is composed of three major subsystems:

1. **Tracking system**: Tracking devices measure the users' current state. This system used WiTilts [75] for tracking the users.
2. **Controller**: The controller provides appropriate corrective feedback based on the leader and follower's state. The controller has to infer the users' motion state to calculate the feedback. My system uses a simple state-based algorithm for inferring a user's current motion-state.
3. **Feedback system**: My system uses Lilypad vibration motors [13] in pulsing mode (refer to chapter 3) to provide motion feedback. The controller provides velocity feedback by changing the frequency of the vibration pulses.

The rest of this section describes the design of the describes the feedback system and the controller in detail.

9.1.1 Feedback system

The motion-synchronization system uses Lilypad vibration motors [13] to provide feedback. The vibration motors provide information about both the particular action to perform as well as the relative speed of the action. The continuous mode (see to chapter 3) is not sufficient to convey both the action and its velocity. Insufficient, in this context, implies using a single motor. Some users suggested using multiple motors to convey both the action and velocity information. Therefore, this system uses the vibration motors in the pulsing mode.

A vibration pulse buzzes the vibration motors for δ ms (for some constant δ). The interval between two vibration pulses signifies the velocity of the motion. From a user's perspective, when the interval between two pulses is reduced, the pulses *appear* quicker. Users naturally associate rapidly occurring pulses with increasing the motion velocity. Similarly, increasing the interval between pulses makes them appear slower, thus signalling the user to slow down.

Ideally, the pulse intervals should be varied continuously with the desired feedback. Practically, humans have a limited ability to distinguish between very similar pulse intervals, especially when the intervals are small (measured in milliseconds). Anecdotal interaction with users suggests that users are especially adept at detecting discrete changes in the pulsing interval.

My system provides four different vibration intervals: 75ms, 150ms, 300ms, 600ms. The feedback provided using these intervals is ordinal; it prompts users to move *faster* or *slower* relative to their current velocity, but does not convey by how much.

Pulsing vibration motors at regular intervals requires an asynchronous feedback system.The motion-synchronization system implements the controller and feedback systems as two separate processes that interact over a socket connection. The feedback system starts a server and waits for messages from the controller. The controller conveys changes in feedback over the socket connection. If the server receives a new message, it acts on the message.

9.1.2 Controller design

The controller interprets the raw sensor data, and calculates the feedback. The motion-synchronization system uses WiTilts [75] that provide the angle of forward and sideways bending for the torso. After calculating the users' state the controller performs two main tasks:

- Infer the users' current motion state from the sensor data.
- Calculate the appropriate action and velocity for the follower.

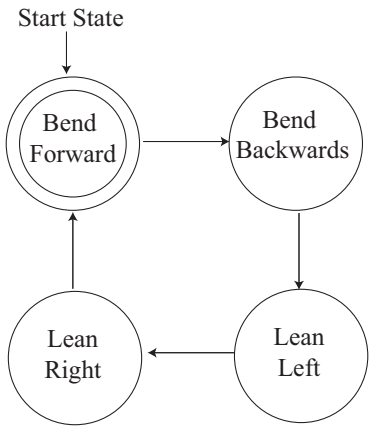


Figure 9.1: An example action represented by four motion segments. The start state is bending forward, and the expected motion consists of cyclically repeating these four motion segments.

Design of the motion-inference system

The controller needs to infer the users' motion state for calculating the appropriate feedback. Sensor noise and the inability of the users to precisely follow motion trajectories makes the problem of inferring the motion state challenging. The motion-synchronization controller uses the state diagram and two other sources of evidence to infer the motion-state:

1. A history of the last N states of the system.
2. The time the user has spent in the current motion state.

The controller is provided a state diagram that describes the expected sequence of motion segments for a given action. Figure 9.1 shows an example of such a state diagram. This state diagram does not specify how long an actor stays in a given motion state.

Figure 9.2 describes the motion-state inference algorithm. The controller calculates two scores, $w(s_i)$ and $T(s_i)$, that estimate the likelihood that the current motion state is s_i. The score $w(s_i)$ is calculated using the history of the user's state. $T(s_i)$ measures the likelihood that a user's motion state is s_i given the motion-state transition diagram, and the time spent in the current motion state. The controller combines these score to determine the final score for the motion state s_i. The controller calculates these two scores for every motion state. The motion state with the maximum score is selected as the current motion state of the user.

$$M_s = \arg\max_i (w(s_i) \cdot T(s_i))$$

The particular forms of the functions $w(s_i)$ and $T(s_i)$ depend on the application. Sections 9.2 provides details of these functions for my system. Conceptually, these scores represent unnormalized probabilities for a given state given the evidence.

Velocity control algorithm

The motion-synchronization system's feedback consists of two parts: prompting the correct motion segment and conveying velocity feedback to match the leader and the follower.

If the leader and the follower are in the same motion state, the controller only needs to determine the velocity for the feedback. However, if their motion states are different the controller must choose

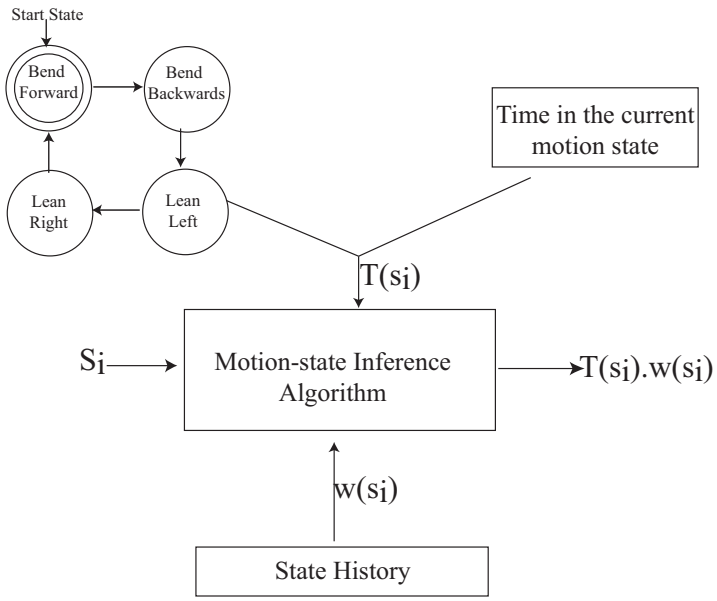


Figure 9.2: A high level view of the motion-state inference algorithm. For a query state s_i, the algorithm calculates two scores $w(s_i)$ and $T(s_i)$. The score $w(s_i)$ calculates the "weight" of s_i based on the history of states for the system. Similarly, $T(s_i)$ calculates the "weight" for the motion state s_i given the motion-state transition diagram and the time the user has been in the current motion state. The final output is the product of the two weights. At any given time, the state with maximum weight is inferred as the current motion state.

between two conflicting objectives: ensuring that the follower follows exactly the same trajectory as the leader, and ensuring that the follower is in the same motion state as the leader.

Figure 9.3 illustrates the conflicting nature of these objectives. The motion sequence for this example consists of two motion segments (and the corresponding motion states) that correspond to "going up" and "going down". In the first part both the leader and the follower are in the same motion state (going up), but the follower is trailing the leader. The controller only determines and conveys the velocity of the motion. In the second part, the leader changes direction and transitions to the "going down" state. At this point, the controller must determine on the direction of motion for the follower. If the primary aim for the follower is to follow the same trajectory as the leader, then the controller should prompt the follower to continue going up (with possibly some velocity feedback). However, if the primary goal of the system is to move the users synchronously, the controller should prompt the follower to change direction to "going down".

For some systems faithfully imitating the leader's trajectory may be the primary objective. However, with a follower who is less responsive to velocity feedback, the two users may get completely out of sync. Further, the delays in the system would keep accumulating even while the users switch between motion states. Measuring the difference between the leader's and the follower's state is more difficult if they are in different motion states. For example, in figure 9.3(b) is it not obvious by how much the follower lags the user. This design choice would pose additional difficulties for controllers that decide the feedback velocity based on this lag.

For some systems, having the leader and follower in the same motion state is more important than imitating the leader's trajectory. For the example shown in figure 9.3, such a controller would direct the follower to the "going down" motion state as soon as the leader transitions to the "going down" state. With this feedback strategy, the two motions would appear more synchronized, which is important for certain applications. Changing the follower's motion state with the leader's

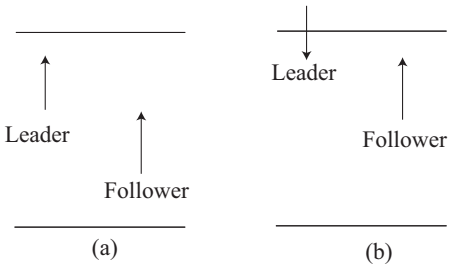


Figure 9.3: The task for both the leader and follower is to periodically going up and down. (a) The leader and the follower are in the same motion state (going up), but the follower trails the leader. (b) The leader changes his motion state to go down. The controller must decide on the feedback for the follower. If the goal is to follow the same trajectory the follower should be asked to continue in the same direction. If the goal is to synchronize the two user, the follower must be asked to change directions as well.

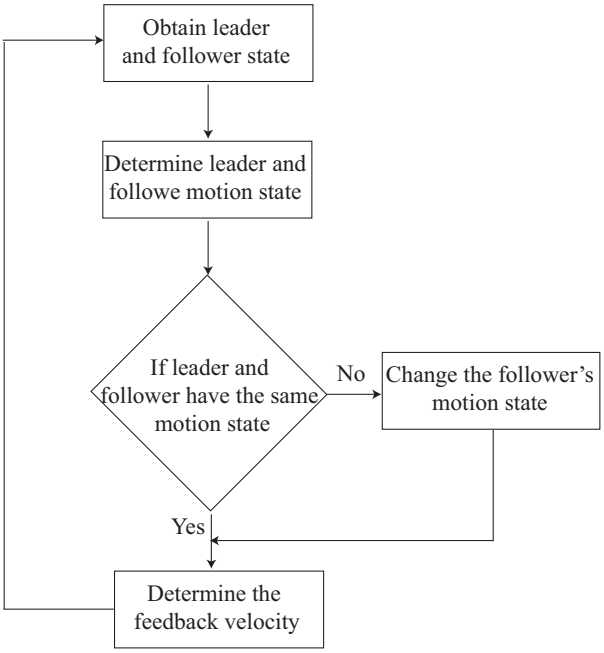


Figure 9.4: A flowchart showing the algorithm for the synchronization system controller.

prevents the accumulation of lag across motion states. Such a system works better when the leader may not perform the same motion consistently across different runs. When there is inconsistency across different runs, a system for exact trajectory following may need to keep additional state information to calculate the feedback. My system implements the second feedback strategy, *i.e.*, ensuring that the actors are in the same state, even if they do not follow the same trajectory.

The controller decides the velocity feedback based on the difference in the state of the leader and the follower. Initially, the controller starts with the normal speed with pulse intervals of 300ms. If the distance in the leader's and follower's state is less than a certain threshold (Δ) the controller pulses the motors at the normal interval. For distances between Δ and 2Δ it reduces the pulse interval to 150ms. For distances beyond 2Δ the pulse interval is further reduced to 75ms. If the follower leads the leader by a distance greater than Δ the pulse interval is increased to 600ms. The controller algorithm is shown in the flowchart in figure 9.4.

9.2 Experiments and results

I tested the motion-synchronization system on 10 volunteers. Section 9.2.1 describes the details of the experiment, the motion tested, and the testing protocol. Sections 9.2.2 and 9.2.3 present the results of the system's effectiveness and user feedback respectively.

9.2.1 Experiment design

I chose a simple repetitive motion to test my system. The motion consists of four motion segments bend forward, bend backward, lean left and lean right. The four motion segments are cyclically repeated for the duration of the experiment. Figure 9.1 shows the state diagram for this example motion.

I used WiTilts mounted on the users' upper back to measure their torso's orientation. WiTilts are three-axis accelerometers that can be used to measure the tilt along its three canonical axes (see chapter 3 for details about WiTilt's working).

Weighting functions for the motion inference algorithm

WiTilts measure the inclination of the back along two axes: one running along the waist, and the other parallel to the ground and perpendicular to the first axis. I will refer to the rotation around the first axis (running along the waist) as the *bend* in the user's torso, and the inclination along the second axis as the *lean*. The bend and the lean angles in the torso corresponds to a user's current state.

A history of such states is used to calculate the weighting function $w(s)$ shown in figure 9.2. The controller uses only the bend angles to estimate the score for the bending forward and bending backwards motion states. Similarly, only lean angles are used to estimate scores for the leaning sideways motion states. This section describes the weighing score, $w(s)$ for the bending motion states; calculating the score for the leaning motion states is similar.

Let N_h be the number of history states used to infer the motion state. Further let D be the total incremental bending during this time:

$$D = \sum_{i=2}^{N_h} |B_i - B_{i-1}|, \tag{9.1}$$

where B_i is the torso bend angle in the ith history state.

Further, let D_f be the magnitude of the change in state when the user is bending forward:

$$D_f = \sum_{j=2}^{N_h} |B_j - B_{j-1}| \ \ \forall j \text{ (s.t. user bends forwards between } j \text{ and } j-1), \tag{9.2}$$

and let n_f be the number of intervals when the user is bending forward. Similarly, define D_b and n_b to be the distance when the user is bending backwards and the number of such intervals. It is easy to measure if the user is bending forwards, or backwards based on the tilt reading.

$$\begin{aligned} n_f + n_b &= N_h - 1 \text{ , and} \\ D_f + D_b &= D \end{aligned}$$

The weights for the two states based on the history are given by

$$W(\text{bending forward}) = \frac{n_f}{N_h - 1} \cdot D_f \text{ and} \tag{9.3}$$

$$W(\text{bending backward}) = \frac{n_b}{N_h - 1} \cdot D_b \ . \tag{9.4}$$

This weighting function rewards a continuous change in the "correct" direction because it increases with n_f (or n_b). If the sensors indicate that the user is consistently moving forward there is a high probability that the user is moving forward. This weighing function also considers the magnitude of the change; a large change in any direction is more likely to come from a movement in that direction rather than due to noise. If both D_f and D_b are small, then both these weighting functions will be small; the user is probably not moving in either direction (the user may be leaning sideways, and those weighting functions would have large values). These properties make this weighting function robust in the presence of noise.

The function $T(s_i)$ assumes that every segment in the motion state contributes at least N_t states to the state history (given the sampling frequency is easy to convert N_t to a minimum duration for every segment). For motions where the user's state does not change too quickly and where every motion segment corresponds to some significant change in the user's state, this assumption is accurate.

Let N_{s_i} be the number of states the user has been in the current state, s_i, at the time of observation, and let the previous and next state in the motion sequence be s_{i-1} and s_{i+1}. Then, $T(s_k)$ for different states s_k in the motion sequence is given by

$$T(s_i) = 1 \tag{9.5}$$

$$T(s_{i-1}) = \begin{cases} \frac{N_t - N_{s_i}}{N_t} & N_{s_i} < N_t \ , \\ \frac{1}{2N_t} & \text{otherwise,} \end{cases} \tag{9.6}$$

$$T(s_{i+1}) = \begin{cases} \frac{N_{s_i}}{N_t} & N_{s_i} < N_t \ , \\ 1 & \text{otherwise,} \end{cases} \tag{9.7}$$

$$T(s_j) = \left\{ \ \frac{1}{2N} \quad j \notin \{i-1, i, i+1\} \ . \right. \tag{9.8}$$

The formula for $T(s_k)$ is based on two principles:

1. A change in a user's motion state is more likely if the user has been in a particular state either too long or too short compared to N_t. If the user has been in a motion state longer than N_t, then the change is a *natural* transition to the next motion state. On the other hand, if the user has been in the current motion state, s_i, for a very short duration, then it is possible that the user never actually transitioned from s_{i-1} to s_i; the erroneous inference was due to noise. As the user spends more time in s_i, the likelihood of the erroneous inference decreases.

2. For every state s_i, a transition to states s_{i+1} or s_{i-1} is much more likely than any other state s_j.

The formula for $T(s_k)$ satisfies the two properties of the weighting function. As the motion transitions from state s_{i-1} to s_i, the weight for s_{i-1} decreases progressively. Similarly, as the user spends more time in state s_i, the weight of transitioning to s_{i+1} increases. At any time during the motion, it is unlikely that the user will skip states in the transition diagram. Hence, the weight for states that are not *adjacent* to the current state is low at all times.

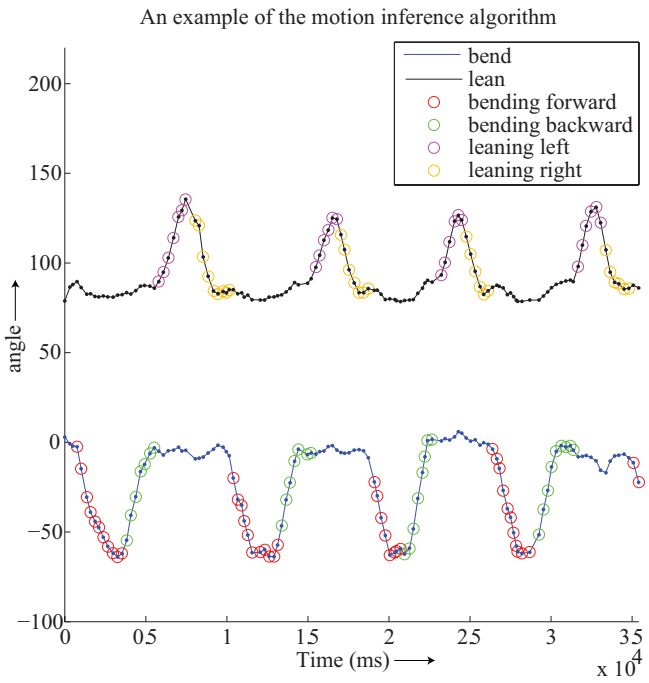


Figure 9.5: An example of the motion difference algorithm. The small blue and black circles show the user's actual state for the bending and leaning angles respectively. The larger circles show the motion segment inference by the algorithm.

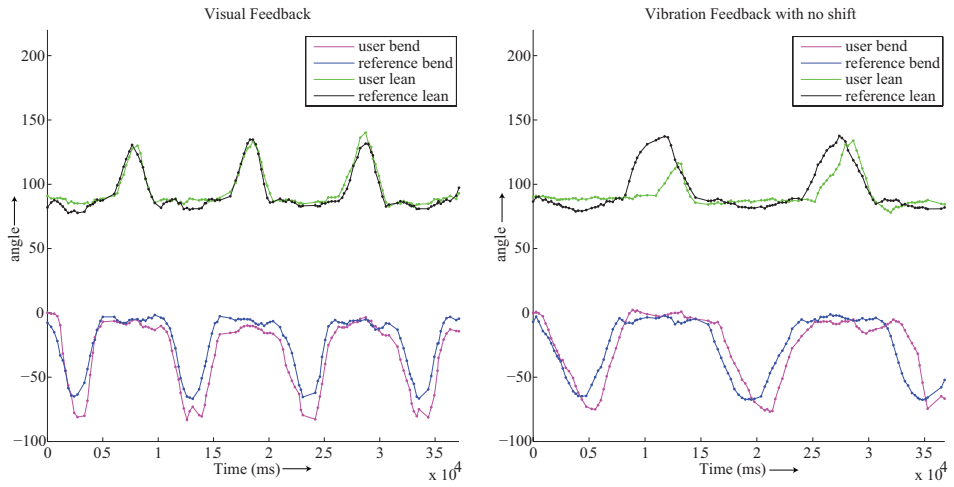
Figure 9.5 shows the results from the motion-inference algorithm. The small blue and black circles show the user's bending and leaning angles respectively. The larger circles show the motion states inferred by the algorithm. For most parts, the algorithm performs very well. Sometimes, the algorithm is a little slow to detect a change in the motion state, because the magnitude of the change in state variables is small. Similarly, the algorithm is robust to noise; the algorithm does not incorrectly classify the local minima in the bend angle at $t \approx 1.25 \times 10^4$ as a change in the motion state.

There is a trade-off in identifying a motion state transition quickly and robustness to noise. The controller may detect changes in motion state faster by querying the WiTilts at a higher frequency. However, at higher sampling frequencies the magnitude of changes in the user's state between successive readings will be small. Therefore, the relative contribution of the sensor's noise will higher.

9.2.2 System results

I tested my system on 10 volunteers. Before the experiment, I described the motion to the volunteers, and showed them an example of the expected motion. The volunteers acted as followers, and followed a leader under three modes of feedback:

- **No feedback**: Both the volunteer and the leader were blindfolded. The leader and the follower both knew the action, but there was no feedback to synchronize their actions.
- **Visual feedback**: The leader was blindfolded, but the follower could see the leader. For a simple motion that we tested, this feedback mode served as the perfect feedback case; the follower could accurately follow the leader with very little lag.



(a) Visual Feedback (b) Feedback from vibration motors

Figure 9.6: Examples of user vs reference trajectories for visual and vibration feedback.

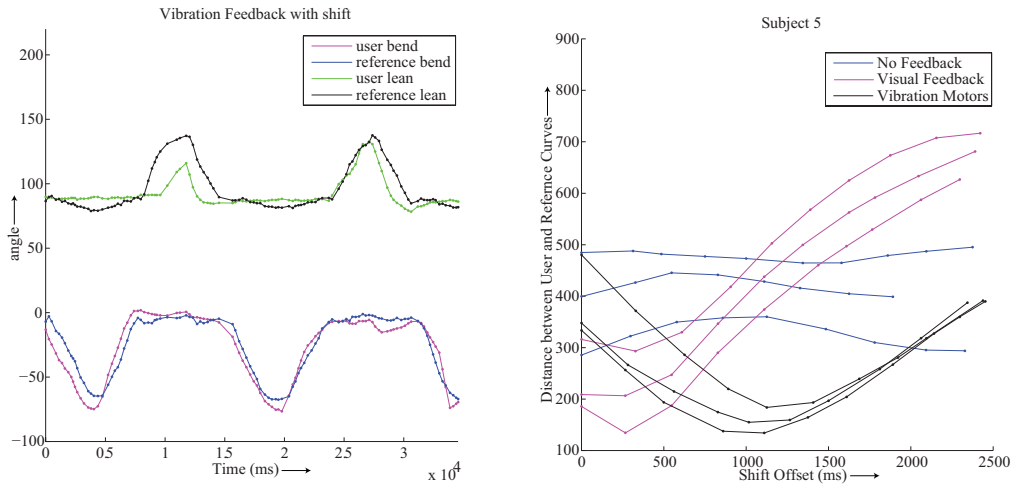
- **Vibration feedback**: Both the leader and the follower were blindfolded, but the follower received synchronization feedback from the motion-synchronization system. This case tests the effectiveness of the my system in providing synchronization feedback.

For each of the three feedback modes, every volunteer was asked to repeat the experiment three times. Every run of the experiment lasted 45 seconds. Both the leader and the follower were calibrated before every run. The calibration process identified the accelerometer's canonical directions, and measured a reference state. All other states were measured relative to this reference state. The leader was blindfolded during all feedback modes to prevent him from synchronizing with the follower.

Figure 9.6 shows the results of one example run each for visual and vibration feedback. Figure 9.6(a) shows the results for the case when the follower could see the reference. As expected, the follower could accurately and closely follow the leader. Such a close matching of the leader's trajectory may not be possible for more complex actions. Figure 9.6(b) shows the follower's performance in response to the vibration feedback that the motion-synchronization system provides. The follower's curve appears *shifted* to the right with respect to that of the leader. This lag is a combination of at least three different factors: lag in inferring a change in the leader's state, lag in conveying the feedback, and the lag introduced by the user in understanding and acting on the feedback. Figure 9.7(a) shows the follower's curve shifted to the left to align with the leader's curve. Such a shift compensates for the lag introduced by various factors described above. The amount of shift needed for the best alignment provides an estimate of the total lag present in the system.

The areas between the users' curves serves as a measure of distance between these curves. As the follower's curve is shifted, the distance between the leader's and follower's curve changes. At some value of the shift this distance becomes minimum. The minimum distance between the two curves provides a measure of how well the system performs when corrected for lag.

Figure 9.7(b) shows the distance between the users' curves for three different runs for all three modes of feedback. Every curve in this figure plots the distance between the users' curves for different shifts of the follower's curve. This figures shows that visual feedback works extremely well; the minimum distance between the curves and the shift required to achieve this minimum are both small. The curves for no feedback shows little change or pattern. For vibration feedback, the



(a) The user reference curve in figure 9.6(b) shifted to the left to align it with the reference curve

(b) shift value vs distance between the curves

Figure 9.7: Shifting the user curve to align with the reference curve. (a) The user curve shifted to achieve optimal alignment between the user and reference curves in figure 9.6(b). (b) The change in distance between the two curves for a user for all three runs for the different feedback methods.

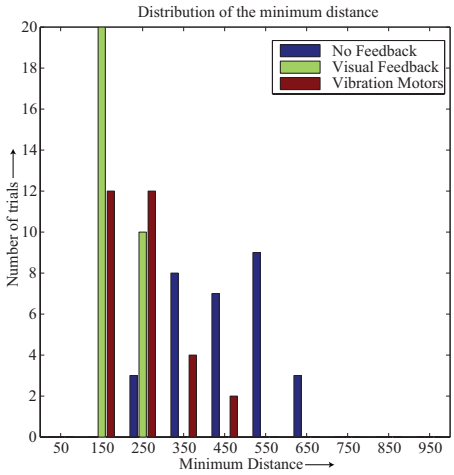
distance between the curves (for all three runs) decreases till it reaches a minimum around 1000 ms, and then increases again. Further, the minimum distance is comparable to that obtained by visual feedback. The shift needed for minimum distance is constant around 1000ms.

Figure 9.8 shows the aggregate results for all the users for all the modes of feedback. Similar to the example in figure 9.7(b), both the minimum distance and the shift needed to achieve the minimum distance are small for visual feedback. For vibration feedback,the distance between the curves is small for the optimal alignment. The shifts needed to optimally align the curves mostly fall in the 1000ms to 1500 ms range. These results imply that my feedback system is consistent both in the quality of optimal synchronization and the lag to achieve the minimum distance. In contrast, both the minimum distance and shift values do not show any clustering when the follower was not provided any feedback.

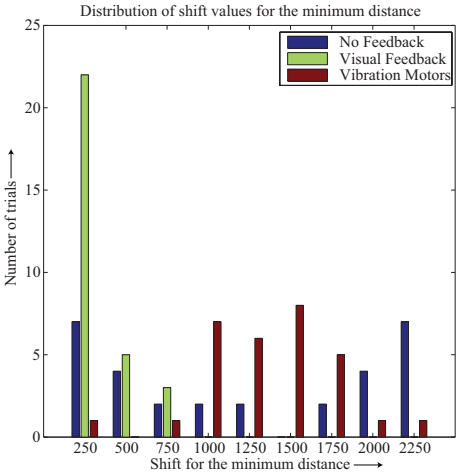
There are many source for the lag in the follower's motion. The current system does not detect changes in the leader's state until after they happen. Empirically, this lag is about 200 ms. Increasing the sampling frequency may reduce this lag. Section 9.2.1 discusses the trade-offs between a higher sampling frequency and the robustness of the motion-inference algorithm. Communicating the feedback to remote XBee modules introduces further lag in the system. Presently, the change in feedback is conveyed to the feedback system server over a socket connection. The server, then conveys the feedback to the local XBee over a Serial connection, which conveys it to the remote XBee over wireless radios. The hysteresis in starting and stopping a vibration motor further adds to the lag. The follower's reaction time is another source of lag. Figure 9.8(b) shows that with visual feedback the lag between the leader and the follower is around 300 ms.

9.2.3 User-experience results

After testing the system, the volunteers were requested to answer a feedback questionnaire. Figure 9.9 shows the volunteer's responses for some of the questions. Seven out of the ten volunteers found the system somewhat easy or extremely easy to learn, whereas two found it somewhat dif-



(a) A histogram of minimum distance between the curves for all runs from all users for the three different types of feedback.



(b) A histogram of shift needed to the user curve to achieve the minimum distance between the curves.

Figure 9.8: A histogram of minimum distance and shift values for the different runs across all users. (a) The minimum distance for all runs for the three modes of feedback. (b) The shift (in ms) needed to optimally align the user curve with the reference curves for all runs for all runs across all users.

ficult (figure 9.9(a)). Similarly, seven volunteers rated the system as somewhat comfortable or extremely comfortable. Eight out of ten volunteers felt the that the level of feedback was inadequate (figure 9.9(c)). However, most users did not have any trouble interpreting the velocity feedback (figure 9.9(d)). Only two volunteers reported that they found interpreting the velocity feedback somewhat difficult.

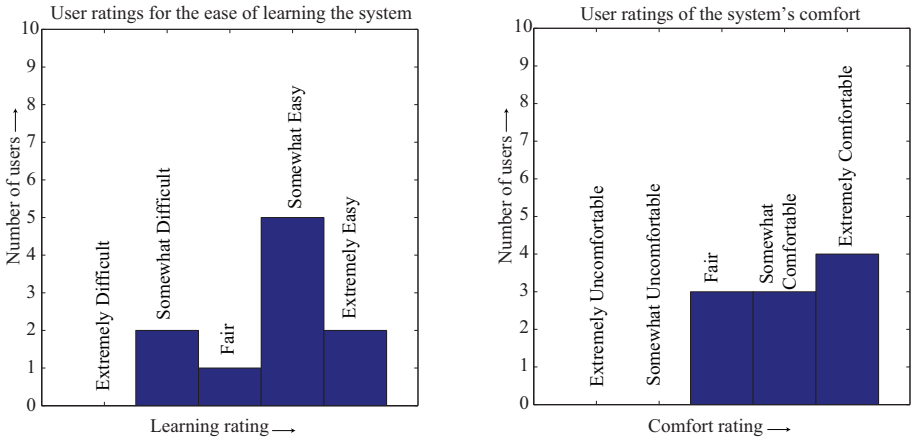
My survey also asked people about the aspects of the system that they liked. Many users liked the concept of providing velocity feedback. However, two users suggested using buzzing intensity instead of pulsing intervals to provide velocity feedback. Using intensity is a novel suggestion, but it suffers from practical problems such as the need for more control pins and higher power requirements. Two volunteers commented that interpreting the velocity feedback required some cognition time, and one volunteer commented that this system made users follow the feedback instead of thinking about the motion task. One user felt that his response to the feedback improved over the course of the experiment. This response suggests that larger testing over longer durations might improve the user performance and satisfaction.

9.3 System limitations and future work

We conducted a small study with only ten volunteers. The system's performance and user response is encouraging. Larger and longer tests with future application should improve such systems.

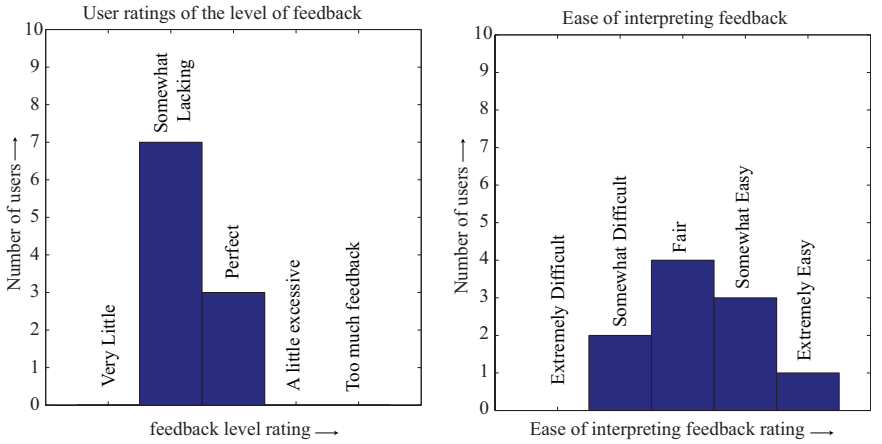
Although the current system enables two users to synchronize their actions, there are two areas of limitations: the lag in tracking the motion and limited feedback. Presently, my system tracks changes in the users' motion state. However, the system can determine the change only *after* the leader has changed state. With more experimental data and detailed models of the underlying motion it may be possible to predict the change in motion state.

The current system changes the frequency to signal an increase or decrease in the motion velocity. This system, however, does not convey the magnitude of the change. There are two sources



(a) User feedback on the ease of learning the feedback

(b) User feedback results on comfort



(c) User feedback results on the level of feedback

(d) User feedback results on the level of feedback

Figure 9.9: Results to questions from a feedback survey administered to the users. (a)How difficult was it to learn the system? (b)How comfortable was the system to use? (c) What was the user's opinion on the level of feedback? (d) How difficult was it to interpret the velocity feedback?

of this limitation: difficulty in conveying accurate feedback with vibration motors, and the limited ability of humans to discern between small changes in the buzzing frequency. Controlling vibration motors over a Bluetooth connection presents many challenges: limitations in accurately spacing the buzzing pulse, start and stop time hysteresis for vibration motors, difficulty of scheduling the feedback process/thread at precise intervals. It may be possible to eliminate some of these limitations by using better vibration motors and smarter microcontrollers to accurately control the buzzing frequency. However, it is not easy to improve a user's ability to identify the changes in pulsing intervals.

An area of future growth for such systems involves more complex motions and more than two actors, *e.g.*, a system that synchronizes many actors as they dance. The arm-motion feedback system described in chapter 8 and the motion synchronization system described in this chapter achieve two complementary objectives. The arm-motion feedback system assists the user to performs an action accurately, but ignores the speed at which the action is performed. The current system synchronizes the motion state of two users, sometimes at the cost of accurately following

the leader's trajectory. It is conceivable to combine the two systems to produce a system that prompts the user to accurately perform a particular action at a specified speed. For example, such a system could be developed in conjunction with exercise videos. A user could receive instructions to follow the video's motion trajectory accurately as well as with the correct speed. Such a system, that augments motion videos, would also solve the problem of finding exemplars for a particular motion.

9.4 Lessons learned

The current system suffers a lag of about one second, which may not be acceptable for some systems. There are at least three different sources of lag in the current system. First, the motion-inference algorithm does not identify a change in motion state until after it has happened. This lag depends on the sampling frequency. The second lag arises due to the delay introduced by the pulsing feedback of the motor. If each vibration pulse lasts for about δ ms, then it may not be possible to convey the change to the user before δ ms. Finally, user reaction time introduces lag in the system. These sources of lag strongly suggest developing predictive systems in the future. With more user motion data, it may be possible to anticipate motion state changes before they occur and possibly provide *predictive* feedback instead of *reactive* feedback.

The performance of the motion-synchronization system is closely related to the limitations of the hardware. Sensor noise limits our ability to quickly identify motion states (refer to section 9.2.1). Similarly, vibration motors could only provide four identifiable velocity states. It is possible to use pulse intensity instead of pulse frequency to convey the change in velocity. This idea also presents practical problems: it would require more XBees because the number of motors would increase, and more power to drive these motors. XBee radios sometimes lost packets. These packet losses could result changes the perceived frequency of the vibration pulses.

Designing and developing the motion-synchronization system required making many design choices and trade-offs. The feedback algorithm involved choosing between synchronizing the users' motion states and matching the users' trajectories accurately. Other examples of trade-offs include the implementation of the feedback system (a separate thread versus a separate process), selecting the sampling frequency of WiTilts (speed of motion state change versus accuracy). With more experimental data and experience it should be easier to understand the costs and benefits of the different alternatives.

Chapter 10

Software Design

This chapter briefly summarizes the software design for the systems developed in this thesis. These systems were developed as C++ applications that run on the Windows platform (XP/Vista/7). Figure 10.1 shows the software architecture for my systems.

System libraries are at the bottom of the software stack. These libraries provide low-level functions such as serial-port communication, threading and socket APIs, etc. Low-level user libraries lie directly above the system libraries. These libraries provide functions that are shared by many applications such as communication with XBee radios, WiTilt modules. These libraries fall under five major categories, which are briefly described in section 10.1. Controllers occupy the next higher level in the software stack. Typically, every new application defines a new controller for that task. A controller comprises different components: the system that it controls, sensors that measure the user's state, etc. These components are implemented in the low-level user libraries. The application code is at the top of the software stack. The application code interacts with controllers as well as different lower-level user libraries. It is responsible for initializing the tracking devices, communication devices, and the controller state. After initializing the different system components, the application code is responsible for calling the library modules in the proper sequence to achieve the system's objectives. Section 10.2 presents the structure of the application code.

I used and modified many free software libraries for my applications. I modified the Vicon real-time SDK library for connecting to and obtaining data from the motion-capture system. Similarly, I ported Andrew Rapp's XBee-Arduino code [85] from Arduino microcontroller's to the Windows platform. This code connects and interacts with remote XBee modules over a serial connection. Porting the XBee code requires type redefinitions that are compatible with the ISO C9x standards, and I used Alexander Chemeris's redefinitions [77].

10.1 Software libraries

Low-level libraries implement many functions that are shared by different applications. I have classified these libraries in the five broad categories. This section describes the design and implementation details for each of these categories.

- **Communication system**: Communications libraries interface with the devices that are used to control the feedback motors: Arduino BT [3] and XBee wireless modules [20]. Communication devices act as a bridge between the controller and the feedback devices (see chapter 3). The communication system implements different feedback strategies used by the controllers. My motion-feedback systems define a standard communicator interface that is shared by different implementation of the communication system. Examples of different implementations

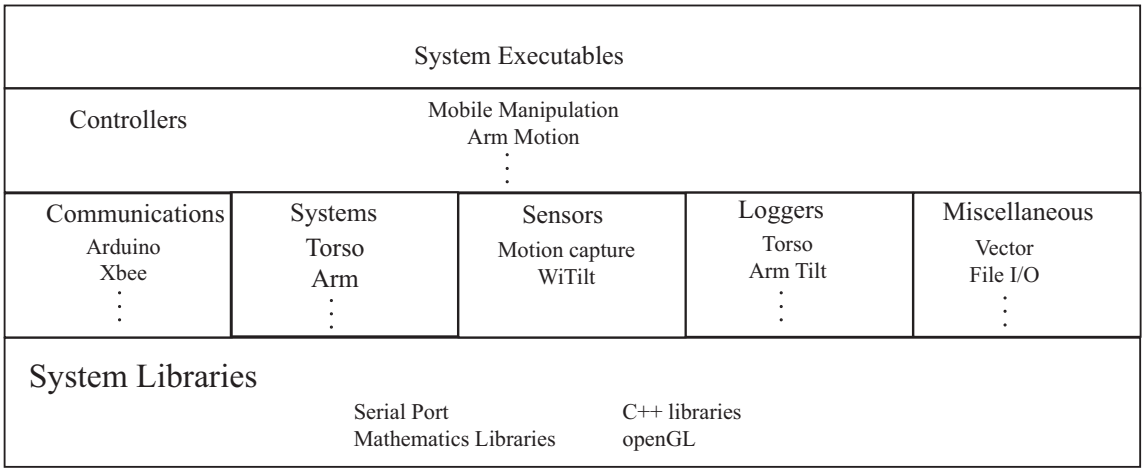


Figure 10.1: The software design for my applications. System libraries are at the bottom followed by user libraries (two layers above the system libraries). At the top are the system applications.

include continuous and pulsing modes for XBee radios, client server based XBee communications, multithreaded XBee communication. A standard interface allowed me to rapidly test different feedback strategies.

- **Control systems**: Typically, the controlled system maintains the user's state in both the workspace and configuration space, and provides interface functions to update the state. Applications update the controlled system's state in the workspace using the interface functions; the update functions internally updates the configuration state. For example, the arm posing system application (chapter 5) only updates the coordinates of the tracked markers on the user's arm, and the update function calculates the angles that define the arm's pose. Many controllers may control the same underlying system. Such a design separates the controlled-system's details from those of the tracking sensors and controllers.

- **Sensor system**: Sensor libraries interface with the sensors that track the user. These library's provide functions necessary to interact with the tracking sensors: establish connections with the sensors, query the user's state, maintain the sensor's state, etc. Every library's function is closely tied to the sensor it interacts with. A sensor library converts the sensor's output to the format used by the controller. For example, the WiTilt library converts raw voltage reading to tilt values.

- **Logging systems**: Logging libraries store and retrieve the system's state during a program's execution. The system's state includes the user's state, the controller's state and the feedback device's state. Logging libraries write the system's state to a file that may be analyzes later. These libraries are primarily used for two purposes: debugging the system and evaluating the system's performance. Although the data that defines the state may vary from one system to another, my logging systems have a standard interface to read and write data to a file.

- **Miscellaneous utilities**: Miscellaneous utilities provide functions that are commonly used by different libraries and applications. Examples of such utilities include libraries that provide functions to manipulate 3D point and vectors, read configuration variables at startup time, low-level file I/O functions, etc.

- **Controllers**: My systems define a new controller for every application. A controller typically

consists of a control system whose state it controls, a communication system to provide the feedback, and functions that store the system's state (using loggers). Every controller implements the logic that determines the correct feedback based on the controlled system's state.

Many controllers share common design attributes. In such cases, my systems define a standard interface that is implemented by different controllers. For example, the controllers for configuration-tasks share the feedback strategy of restricted feedback. Every controller that provides restricted-feedback implements the interface for restricted-feedback systems.

10.2 System applications

Libraries provide the functionalities required to track the user, determine the feedback, and convey the feedback to user. An application uses these functions towards a particular goal. Every new system is developed using a separate application.

An application's functioning is partitioned in two distinct phases: initialization and control loop. The control loop is an iterative process that tracks the user and provide corrective feedback.

An application, typically, performs the following initialization tasks during startup:

- **Read configurations settings**: An application has several attributes that change from one computer to another, or between different runs of the same program. Some examples of such attributes are the files used to log the system's data, serial ports used for communications, feedback mode for XBees, address of an XBee server, etc. These configurable properties are written to a settings file before starting the application. During initialization the application reads these properties from the settings file.

- **Initialize communication with the sensors and feedback devices**: It is necessary to establish a connection with the tracking and communications devices before using them to track the user and provide him feedback. The application needs to establish a socket connection with the motion-capture system before streaming data. Similarly, before communicating with WiTilts, Arduino boards and XBee modules the application must obtain the serial port communicating with these devices. Typically, the configuration information needed to establish the connection, such as the serial-port address, are read from a configuration file.

- **Initialize the controlled system and system loggers**: The relevant part of the user's body, *e.g.*, the arms constitute the controlled system. Before using the controller functions to track and provide feedback the application must initialize the controlled system. Initializing the system involves measuring and updating the user's state for the first time. For example, initializing the arm for the arm-posing system (chapter 5) involves calculating the lengths of the arm and forearm, and updating the initial configuration of the arm.

  Similar to controlled system, the application initializes the appropriate logger. The log file's details are usually provided in a configuration file.

- **Initialize the application controller**: My systems typically define a new controller for every application. The controller determines and conveys the feedback based on the controlled system's state. During initialization, the application associates the different components of the system with the application's controller. Examples of the components associated with the controller include the controlled system, the communication system, and loggers. For C++ application associating components implies passing a reference to the underlying objects.

Another important step in initialization involves specifying the controller's objective. For configuration tasks, the objective is the target configurations. Target configurations for the mobile-manipulation system and the arm-posing system are provided during initialization. On the other hand, the shirt-folding system calculates the target configurations at run time.

The motion objectives for trajectory tasks depend on the system. The ideal bending and leaning angles and their acceptable error thresholds are the objectives for the posture system. A sequence of foreground and background tasks are the objectives for the arm-motion feedback system. Each motion-task specifies the target configuration for the arm, and the time to wait before moving to the next motion task. For the motion-synchronization system, the state-diagram of motion-segments defines the motion objective. The application controller is initialized with the motion-objectives during the initialization step.

At the end of the initialization step all components of the application are ready for providing motion feedback. During the control-loop step, the controller runs in a loop where it continuously performs three function till the program terminates:

- **Read the sensor data**: The application periodically queries the tracking sensors. The frequency of these updates varies between applications. The sensor libraries provide interface functions for querying the user's state. For sensors, such as WiTilts, that buffer the sensor data, sensor libraries ensure that the applications are not provided "stale" data.

- **Updating the controlled system's state**: Applications process the sensor output to extract the relevant data. The application updates the controlled-system's state using controlled system's interface functions.

  For example, in the shirt-folding system the motion-capture system outputs the positions of 11 markers that define the user skeleton. The shirt-folding system extracts the data for the six relevant markers (shoulder, elbow, and index finger for each arm), and updates the controlled system.

- **Apply feedback**: Typically a controller provides an interface function for applying feedback. This function calculates the required feedback based on the controlled-system's current state and the controller's motion objective. If the controller determines any change in the necessary feedback, the appropriate feedback is conveyed to the user.

  After updating the controlled-system's state, the application prompts the controller to provide the relevant feedback based on the user's new state.

Chapter 11

Future Work

Systems that track humans and provide realtime feedback and relatively new. This chapter aims to provide both specific applications for future research as well as to highlight new areas of investigation. Some future research areas mentioned in this chapter extend those already covered in this thesis, whereas others address completely new problems. This chapter discusses the future of human motion feedback systems from two different perspectives: design of future systems and potential applications.

Mobile computation platforms, such as smart phones, and newer, cheaper, and more accurate tracking- and feedback-devices will impact the design of future systems. Section 11.1 discusses the potential designs of future systems. Section 11.2 presents possible applications for everyday motion tasks. Gait measurement and correction is an important area for medical and sports applications, and sections 11.3 discusses some possible ways the current systems can be extended to these domains. Sections 11.4 presents potential applications for personal health monitoring systems respectively with emphasis on rehabilitation and passive tracking. Teaching new motions is an important goal for several applications. Section 11.5 presents the essential components of a motion training system. My systems for configuration tasks employ restricted feedback. Section 11.6 describes future areas of research for approximating trajectories using restricted feedback.

11.1 Future extensions

My systems use user-mounted sensors that relay the user's state to a computer. The controller software that runs on the computer analyzes the tracking data and provides feedback over a wireless connection. Although these three components –sensors, controller and feedback devices– are integral to every controlled system, their design and implementation changes with applications. In this section, I will present some possible designs for future systems.

My systems need a computer, which is either a desktop or a laptop, for running the application software. In environments such as open fields, courts, or gymnasiums such systems have limited portability. New mobile computational devices such as smart-phones and music players offer alternative platforms that address this limitation. Instead of running on a computer, future applications may run the controller program on a mobile device such as a smart-phone. These devices are smaller, more portable, and consume less power than traditional computers. At present, these devices offer limited connectivity with peripheral devices such as external sensors, but I expect the connectivity to improve in the future.

Another alternative platform for running the controller could be a microcontroller board. Microcontroller boards offer the advantages of cheaper cost, direct interfacing with external devices, and

lower power usage. They can be ideal for applications that are not computationally intensive such as the posture shirt described in chapter 7. The major drawback for microcontroller-board-based systems is limited user-interface and difficulty in customizing applications.

The cost-effectiveness, size, and accuracy of the sensors that I used (accelerometers and motion-capture system) has rapidly improved in recent years. Other sensors such as compasses, IMUs, gyroscopes, SONAR are improving along similar lines. Many of these sensors are designed as MEMS devices that offer standard software interfaces such as analog or I2C [56]. These sensors could be utilized as tracking devices in the future. One example of such a system is the posture chair developed by Zheng *et al.* [90], which uses force sensitive resistors for detecting the user's posture.

I have developed my application using vibration motors as feedback devices. There are other possible modes of feedback such as auditory, visual, etc. The right feedback device depends on the application.

The tracking sensors, computational devices and feedback device technologies are changing rapidly. These technologies are converging towards a standard interface. In the future, motion feedback systems should get cheaper, more portable and easier to develop and deploy.

11.2 Practical everyday applications

Many practical, everyday tasks can benefit from motion-feedback systems. This section describes three such applications: exercise training, yoga, and riding a bicycle. Users are *inside* the system during these activities, and so they cannot easily determine their mistakes. Tracking users performing these activities with external sensors is straightforward. Furthermore, vibration motors provide the ideal mode of feedback, because unlike other modes of feedback, such as visual feedback, they do not distract the user while performing the motion. These three applications are specific examples, and I speculate than the current systems could be extended to several other potential applications.

Many forms of physical exercise such as calisthenics or weight training can benefit from motion-feedback systems. The arm-motion feedback system (chapter 8) demonstrates the potential for such applications. These forms of physical exercise require good form, multiple repetitions, and timing to perform them adequately. A motion-feedback system can assist users to improve all these attributes.

Yoga positions often require users to maintain a particular pose. An application that extends the posture shirt (chapter 7) to other parts of the body may help users improve their yoga exercise routine.

Both the physical exercise applications and yoga-feedback applications will benefit from alternate system designs. Markerless optical tracking systems such Microsoft Kinect [42] provide easier and cheaper tracking. Similarly, controllers running on smart phones or music players will enable users to "carry" these systems to exercise areas such as gymnasiums.

While riding the bicycle or a unicycle the rider must balance the center of gravity. If the center of gravity moves beyond a threshold on either side, the rider cannot continue to cycle, and falls down. Beginners may find it hard to gauge their balance, and external sensors such as accelerometers and gyroscopes along with corrective feedback can help them maintain their balance.

Many riders use training wheels to prevent injuries from falling down. These wheels often prove inefficient because riders continuously lean on them. Adding bump-sensors to the training wheels may produce a system that reduce this leaning on the training wheels: if the sensors detect that a training wheel is touching the ground, the controller buzzes the user to move in an appropriate

manner.

11.3 Gait measurement and correction

Neurological diseases, such as Parkinson's disease, affect a patient's gait. The effect on a patient's gait and the underlying cause depends on the disease. For example, Parkinson's patients exhibit a reduced walking velocity ([24], [73]) due to a shortening of the stride length. Some diseases display multiple symptoms and multiple diseases may share the same gait disorder [78]. Typically, a patient must visit a doctor's clinic to diagnose a disease, evaluate the disease's progress and any improvements caused by specific treatments. Systems such as the one developed in this thesis can measure the patient's gait parameters in a natural environment. Keijsers *et al.* [41] survey systems that use wearable sensors (mainly accelerometers and gyroscopes) to measure dyskinesia and tremor in patients with Parkinson's disease. Stolze *et al.* [78] have shown that acoustic or visual cues (metronomes and stripes on the ground) improve gait parameters for patients with Parkinson's disease and those suffering from normal pressure hydrocephalus. Vibration feedback, which is more intuitive and less intrusive than audio or visual cues, may prove a better alternative for providing corrective feedback.

Runners, especially long-distance runners, frequently need feedback about gait parameters such as stride length, foot pressure, speed, etc. It is possible to measure many of these parameters with body mounted sensors [44]. By interfacing these user-mounted sensors with mobile computation platforms such as music players, it may be possible to provide real-time feedback that corrects various gait parameters in runners.

11.4 Rehabilitation and passive tracking

Passive tracking refers to tracking people with sensors placed in the environment instead of mounting them on the user's body. For example, motion detectors switch of lights if they do not detect any activity in a room. Passive tracking may prove useful for tracking patients or elderly people staying alone. Sensors in the environment may provide information about people's daily activities such as eating or sleeping. Patients, their family, and care-givers may use such activity data for their specific need. For example, a system that reminds user to take medication can help users maintain their daily medication schedule.

Recovery from stroke or trauma often requires physical therapy. During the recovery process, patients are asked to repeatedly perform a set of exercises. Many existing haptic systems such as exoskeleton address this problem (see to chapter 2). However, for motions that are easy to track and correct, systems such as the arm-motion feedback system described in chapter 8 provide a simpler and less intrusive alternative. Such systems can also improve the efficiency of simple exercises. For example, one rehabilitation exercise requires users to squeeze a rubber ball. By placing force sensors in the ball and timing the user's action the motion-feedback system can ensure that the user squeezes the ball with adequate force and with the right frequency.

11.5 Teaching new motions

For many applications, the ultimate goal is to either teach a new motion, or increase the speed of learning by providing real-time, corrective feedback. Teaching new motions was not a goal for this thesis, but it is an important area of future investigation.

The systems presented in this thesis enable users to attain or improve their performance on a motion objective. However, these systems do not teach new motions.

Testing an application's teaching potential requires careful experiment planning over a long duration. Investigators must demonstrate several different results to prove an application's ability to teach a new motion. First, the gains from motion feedback should persist long after the feedback is removed. Next, users that received feedback must attain their true potential quicker than control subjects that did not receive the feedback. Finally, after learning the motion, users that receive the feedback must perform the action as well or better than those that did not receive any feedback.

Some applications developed as part of this thesis and others suggested in this chapter would benefit from long-term studies that show their teaching potential. The posture shirt (chapter 7), and the arm motion controller (chapter 8) are examples of applications from this thesis that would benefit from studies that demonstrate their teaching potential.

11.6 Algorithms for restricted feedback

The applications for configuration-tasks provide restricted feedback. Restricted feedback limits the possible paths that the controlled-system can follow through configuration space. Therefore, users can only approximate complex motions when provided restricted feedback. Restricted feedback can be useful for other robotics systems. For example, a multiple-degrees-of-freedom robot arm that can only articulate one (or a few) degrees of freedom at a time due to power consideration would employ restricted feedback.

Chapter 2 discussed the similarity of approximating trajectories with the polygonal-approximation (PA) problem. The PA algorithms approximate a given curve with a sequence of lines that minimize the Euclidean distance between the actual curve and the approximating lines. It is possible to define other optimization criterion for other practical considerations such as the number of motion-segments, average length of a motion-segment, etc. Obstacles in the workspaceprovide further limit the allows trajectories. Extending algorithms from approximating a motion path to calculating a motion path (between a source and a target configuration) with restricted feedback (in presence of obstacles) could provide newer, simple and elegant motion-planning algorithms

Acknowledgments

This thesis would not have been possible without the help and support of many, many people.

I would like to thank my parents for their love and support. Right from my childhood they always encouraged me to make my own decisions, but were available when I needed advise. Without their encouragement, I would not have decided to pursue a Ph.D. degree.

My wife, Priya, has always been a constant source of motivation and encouragement. She has supported me as I moved from industry to academia and back to the industry, and as I jumped from one research project to another. Her love and faith in my abilities kept me going during the frustrating times of my research.

I also owe many thanks to my advisor, Devin Balkcom. Devin provided me with the freedom to explore many areas in Computer Science and Robotics. I did not start on my thesis problem till my fifth year, something that would not have happened without his patience.

Many thanks to Katsu Yamane, Andrew Campbell, and Chris Bailey-Kellogg for valuable discussions and suggestions as I worked my way through this thesis. I was fortunate to closely collaborate with Chris Bailey-Kellogg for a significant period at Dartmouth. Chris's insistence on conducting more experiments and analyzing data from all possible perspectives were critical in my growth as a computer scientist.

During my long stay at Dartmouth, I have interacted and made friends with a lot of wonderful people. Many thanks are due to my lab mates–Matt, Anne, Andrei, and Govind–who were a source of great ideas and were always willing to help me. Special thanks to Sphurti Saraph for helping me build and test the first few prototype systems. Also, thanks are due to the many friends I made at Dartmouth: Raz, Chien-Chung, Umang, Ranganath, and Shrirang, the pot lucks and board games were critical for sustaining me during my stay at Dartmouth.

The CS department staff has been exceptionally helpful all through these years. I owe a lot of gratitude to Emily, Joe, Christine and Kelly, who were always ready to help me resolve the administrative issues. Last, but not the least, I would like to thank Wayne, Tim, and Sandy who not only kept my computers running, but also always accommodated my random and sometimes outrageous requests about new hardware and software.

This work was supported in part by the grant NSF CAREER (0643476).

Bibliography

[1] A.U. Alahakone and S.M.N.A. Senanayake. Vibrotactile feedback systems: Current trends in rehabilitation, sports and information display. In *Advanced Intelligent Mechatronics, 2009. IEEE/ASME International Conference on*, pages 1148 –1153, July 2009.

[2] G. Alexander, T. Havens, M. Skubic, M. Rantz, J. Keller, and Abbott C. Markerless human motion capture-based exercise feedback system to increase efficacy and safety of elder exercise routines. *Gerontechnology*, 7(3):67, May 2008.

[3] Arduino BT product website, 2010. `http://arduino.cc/en/Main/ArduinoBoardBluetooth`.

[4] Atmel Corporation, 2325 Orchard Parkway, San Jose, CA 95131, USA. *ATMEL 8-bit AVR Microcontroller*, Jul 2010.

[5] E.R. Bachmann, Xiaoping Yun, D. McKinney, R.B. McGhee, and M.J. Zyda. Design and implementation of MARG sensors for 3-DOF orientation measurement of rigid bodies. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 1, pages 1171–1178 vol.1, Sept. 2003.

[6] Jernej Barbič, Alla Safonova, Jia-Yu Pan, Christos Faloutsos, Jessica K. Hodgins, and Nancy S. Pollard. Segmenting motion capture data into distinct behaviors. In *In Graphics Interface*, pages 185–194, 2004.

[7] Faisal Bashir, Ashfaq Khokhar, and Dan Schonfeld. A hybrid system for affine-invariant trajectory retrieval. In *Proceedings of the 6th ACM SIGMM International Workshop on Multimedia Information Retrieval* , pages 235–242. ACM New York, NY, USA , 2004.

[8] Matthew Bell and Devin J. Balkcom. Grasping Non-stretchable Cloth Polygons. *The International Journal of Robotics Research*, 29(6):775–784, May 2010.

[9] Richard Bellman. On the approximation of curves by line segments using dynamic programming. *Communications of the ACM*, 4(6):284, June 1961.

[10] Bluegiga technologies, Bluegiga Technologies Oy, Sinikalliontie 5 A, 02630 Espoo, Finland. *WT11 Bluetooth Module Description*, Jan 2009.

[11] J. Borenstein. The navbelt - a computerized multi-sensor travel aid for active guidance of the blind. In *CSUN's Fifth Annual Conference on Technology and Persons with Disabilities*, pages 107–116, 1990.

[12] Matthew Brand and Aaron Hertzmann. Style machines. In *Proceedings of SIGGRAPH 2000*, pages 183–192, 2000.

[13] L. Buechley. A construction kit for electronic textiles. In *Wearable Computers, 2006 10th IEEE International Symposium on*, pages 83–90, Oct. 2006.

[14] W.S. Chan and F. Chin. Approximation of polygonal curves with minimum numbers of line segments or minimum error. *International Journal of Computational Geometry and Applications*, 6:59–77, 1996.

[15] William Chen and Shih fu Chang. Motion trajectory matching of video objects. In *Proceedings of the 6th ACM SIGMM international workshop on Multimedia information retrieval* , pages 235–242, 2000.

[16] Michelle Clifford and Leticia Gomez. *Measuring Tilt with Low-g Accelerometers*. Freescale Semiconductor, Inc., Freescale Semiconductor Inc., Technical Information Center, CH370, 1300 N. Alma School Road, Chandler, Arizona 85224, May 2005.

[17] F.S. Cohen, Zhaohui Huang, and Zhengwei Yang. Invariant matching and identification of curves using B-splines curve representation. *Image Processing, IEEE Transactions on*, 4(1):1–10, Jan 1995.

[18] Copying arm poses application, 2010. `http://www.cs.dartmouth.edu/~robotics/movies/pak-movies/copyPose03ClipWithCaptions.mov`.

[19] A. Cullell, J.C. Moreno, and J.L. Pons. The Gait Orthosis. A robotic system for functional compensation and biomechanical evaluation. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3136 –3137, April 2007.

[20] Digi International Inc., 11001 Bren Road East, Minnetonka, MN 55343. *XBee®/XBee-PRO®RF Modules*.

[21] R.L. Scot Drysdale, Günter Rote, and Astrid Sturm. Approximation of an open polygonal curve with a minimum number of circular arcs and biarcs. *Computational Geometry*, 41(1-2):31 – 47, 2008. Special Issue on the 22nd European Workshop on Computational Geometry (EuroCG), 22nd European Workshop on Computational Geometry.

[22] James George Dunham. Optimum uniform piecewise linear approximation of planar curves. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8(1):67–75, Jan. 1986.

[23] S. S. Everett, Wauchope K., and M. A. Prez Quiones. Creating natural language interfaces to VR systems. *Virtual Reality*, 4(2):103–113, 1999.

[24] M. Faist, J. Xie, D. Kurz, W. Berger, C. Maurer, P. Pollak, and C. H. Lcking. Effect of bilateral subthalamic nucleus stimulation on gait in Parkinson's disease. *Brain*, 124(8):1590–1600, 2001.

[25] Elisabetta Farella, Luca Benini, Bruno Ricc, and Andrea Acquaviva. "moca: A low-power, low-cost motion capture system based on integrated accelerometers". *"Advances in Multimedia"*, 2007, 2007.

[26] David J. Fleet, Michael J. Black, Yaser Yacoob, and Allan D. Jepson. "design and use of linear models for image motion analysis". *"International Journal of Computer Vision"*, 36(3):171–193, 2000.

[27] Ajo Fod, Maja J Mataric, and Odest Chadwicke Jenkins. Automated derivation of primitives for movement classification. *Autonomous Robots*, 12:39–54, 2002.

[28] Freescale Semiconductor Inc., Technical Information Center, EL516, 2100 East Elliot Road, Tempe, Arizona 85284. *MMA7260Q XYZ Three-Axis Low g Acceleration Sensor*, 2005.

[29] Freescale Semiconductor Inc., Technical Information Center, EL516, 2100 East Elliot Road, Tempe, Arizona 85284. *±1.5g, ±6g Three Axis Low-g Micromachined Accelerometer*, 2008.

[30] K. Hachimura, H. Kato, and H. Tamura. A prototype dance training support system with motion capture and mixed reality technologies. In *Robot and Human Interactive Communication, 2004. 13th IEEE International Workshop on*, pages 217–222, Sept. 2004.

[31] P.R.G. Harding and T. Ellis. Recognizing hand gesture using fourier descriptors. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 3, pages 286–289 Vol.3, Aug. 2004.

[32] Joel A. Hesch and Stergios I. Roumeliotis. Design and analysis of a portable indoor localization aid for the visually impaired. *International Journal of Robotics Research*, 29(11):1400–1415, Sept. 2010.

[33] N. Hogan, H.I. Krebs, J. Charnnarong, P. Srikrishna, and A. Sharon. MIT-MANUS: a workstation for manual therapy and training. In *Robot and Human Communication, 1992. Proceedings., IEEE International Workshop on Robot and Human Communication*, pages 161–165, Sept. 1992.

[34] M.K. Holden, T.A. Dyar, and L. Dayan-Cimadoro. Telerehabilitation using a virtual environment improves upper extremity function in patients with stroke. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 15(1):36–42, March 2007.

[35] Ming-Kuei Hu. Visual pattern recognition by moment invariants. *Information Theory, IEEE Transactions on*, 8(2):179–187, Feb 1962.

[36] H. Imai and M. Iri. Polygonal approximation of a curveformulations and algorithms. *In: G.T. Toussaint, Editor, Computational Morphology*, pages 71–86, 1988.

[37] J. Lei J. Maitin-Shepard, M. Cusumano-Towner and P. Abbeel. Cloth Grasp Point Detection based on Multiple-View Geometric Cues with Application to Robotic Towel Folding. In *IEEE International Conference on Robotics and Automation*, 2010.

[38] D. Jack, R. Boian, A.S. Merians, M. Tremaine, G.C. Burdea, S.V. Adamovich, M. Recce, and H. Poizner. Virtual reality-enhanced stroke rehabilitation. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 9(3):308–318, Sept. 2001.

[39] Lewis Johnson, Jeff Rickel, Randy Stiles, and Allen Munro. Integrating pedagogical agents into virtual environments. *Presence: Teleoperators and Virtual Environment*, 7:523–546, 1998.

[40] Michael Patrick Johnson, Andrew D. Wilson, Bruce Blumberg, Christopher Kline, and Aaron F. Bobick. Sympathetic interfaces: Using a plush toy to direct synthetic characters. In *ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 152–158, 1999.

[41] N.L.W. Keijsers, M.W.I.M. Horstink, and S.C.A.M. Gielen. Online monitoring of dyskinesia in patients with Parkinson's disease. *Engineering in Medicine and Biology Magazine, IEEE*, 22(3):96 –103, May-June 2003.

[42] Kinect-xbox.com, 2011. http://www.xbox.com/en-US/kinect.

[43] Lucas Kovar, Michael Gleicher, and Frederic Pighin. Motion graphs. In *Proceedings of SIGGRAPH 2002*, pages 473–482, 2002.

[44] Jihong Lee and Insoo Ha. Sensor fusion and calibration for motion captures using accelerometers. In *Robotics and Automation, 1999. IEEE International Conference on*, volume 3, pages 1954–1959 vol.3, 1999.

[45] Jung-Ah Lee, Sang-Hyun Cho, Jeong-Whan Lee, Kang-Hwi Lee, and Heui-Kyung Yang. Wearable accelerometer system for measuring the temporal parameters of gait. *Conf Proc IEEE Eng Med Biol Soc*, 2007:483–6, 2007.

[46] J.. Lieberman and C.. Breazeal. Tikl: Development of a wearable vibrotactile feedback suit for improved human motor learning. *IEEE Transactions on Robotics*, 23(5):919–926, Oct. 2007.

[47] Manipulation application, 2009. http://www.cs.dartmouth.edu/~robotics/movies/pak-movies/RightArmPlace07.mov.

[48] Holden Maureen, Todorov Emanuel, Callahan Janet, and Bizzi Emilio. Virtual environment training improves motor performance in two patients with stroke: Case report. *Journal of Neurologic Physical Therapy*, 23(2):57–67, May 1999.

[49] F. Mokhtarian and A.K. Mackworth. A theory of multiscale, curvature-based shape representation for planar curves. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(8):789–805, Aug 1992.

[50] Farzin Mokhtarian, Sadegh Abbasi, and Josef Kittler. Robust and efficient shape indexing through curvature scale space. In *In Proceedings of British Machine Vision Conference*, pages 53–62, 1996.

[51] Javier Mora, Won-Sook Lee, and Gilles Comeau. 3D visual feedback in learning of piano posture. In *Technologies for E-Learning and Digital Entertainment*, volume 4469 of *Lecture Notes in Computer Science*, pages 763–771. Springer Berlin / Heidelberg, 2007.

[52] Motion reality golf system, 2010. http://www.motionrealitygolf.com/products.html.

[53] Navigation and manipulation application, 2009. http://www.cs.dartmouth.edu/~robotics/movies/pak-movies/NavPickAndPlace02.mov.

[54] Navigation application, 2009. http://www.cs.dartmouth.edu/~robotics/movies/pak-movies/NavMovie09.mov.

[55] Northstar systems, 2011. http://www.evolution.com/products/northstar/.

[56] NXP Semiconductors. *I2C-bus specification and user manual*, 03 edition, Jun 2007.

[57] Costas Panagiotakis and George Tziritas. Any dimension polygonal approximation based on equal errors principle. *Pattern Recognition Letters*, 28(5):582 – 591, 2007.

[58] T. Pavlidis and S.L. Horowitz. Segmentation of plane curves. *Computers, IEEE Transactions on*, C-23(8):860–870, Aug. 1974.

[59] Juan-Carlos Perez and Enrique Vidal. Optimum polygonal approximation of digitized curves. *Pattern Recognition Letters*, 15(8):743 – 750, 1994.

[60] J.C. Perry, J. Rosen, and S. Burns. Upper-limb powered exoskeleton design. *Mechatronics, IEEE/ASME Transactions on*, 12(4):408 –417, Aug. 2007.

[61] A. Pikaz and I. Dinstein. Optimal polygonal approximation of digital curves. In *Pattern Recognition, 1994. Vol. 1 - Conference A: Computer Vision & Image Processing., Proceedings of the 12th IAPR International Conference on*, volume 1, pages 619–621, Oct 1994.

[62] L. Piron, P. Tonin, F. Cortese, M. Zampolini, F. Piccione, M. Agostini, C. Zucconi, A. Turolla, and M. Dam. Post-stroke arm motor telerehabilitation web-based. In *Virtual Rehabilitation, 2006 International Workshop on*, pages 145–148, 0-0 2006.

[63] Lamberto Piron, Paolo Tombolini, Andrea Turolla, Carla Zucconi, Michela Agostini, Mauro Dam, Giovanna Santarello, Francesco Piccione, and Paolo Tonin. Reinforced feedback in virtual environment facilitates the arm motor recovery in patients after a recent stroke. In *Virtual Rehabilitation, 2007*, pages 121–123, Sept. 2007.

[64] Poses to pick objects from a box, 2010. `http://www.cs.dartmouth.edu/~robotics/movies/pak-movies/RightArmPose05.mov`.

[65] Ramesh Raskar, Hideaki Nii, Bert deDecker, Yuki Hashimoto, Jay Summet, Dylan Moore, Yong Zhao, Jonathan Westhues, Paul Dietz, , John Barnwell, Shree Nayar, Masahiko Inami, Philippe Bekaert, Michael Noland, Vlad Branzoi, and Erich Bruns. Prakash: lighting aware motion capture using photosensing markers and multiplexed illuminators. In *ACM SIGGRAPH 2007*, volume 36, 2007.

[66] Bimal Kr. Ray and Kumar S. Ray. A non-parametric sequential method for polygonal approximation of digital curves. *Pattern Recognition Letters*, 15(2):161–167, 1994.

[67] Jeff Rickel and W. Lewis Johnson. Animated agents for procedural training in virtual reality: Perception, cognition, and motor control. *Applied Artificial Intelligence*, 13:343–382, 1998.

[68] Rn41 Bluetooth module from Roving Networks, 2010. `http://www.rovingnetworks.com/rn-41.php`.

[69] Marc Salotti. An efficient algorithm for the optimal polygonal approximation of digitized curves. *Pattern Recognition Letters*, 22(2):215 – 221, 2001.

[70] D.M. Shawver. Virtual actors and avatars in a flexible user-determined-scenario environment. In *Proceedings of IEEE VRAIS*, pages 170–177, 1997.

[71] Shirt folding application, 2010. `http://www.cs.dartmouth.edu/~robotics/movies/pak-movies/shirtFolding1.mov`.

[72] Shraga Shoval, Johann Borenstein, and Yoram Koren. Auditory guidance with the navbelt - a computerized travel aid for the blind. *IEEE Transactions on Systems, Man, and Cybernetics*, 28:459–467, 1998.

[73] Karen Lohmann Siegel and Leo Verhagen Metman. Effects of bilateral posteroventral pallidotomy on gait of subjects with Parkinson disease. *Arch Neurol*, 57(2):198–204, 2000.

[74] Ronit Slyper and Jessica Hodgins. Action capture with accelerometers. In *2008 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, July 2008.

[75] Sparkfun Electronics, 6175 Longbow Drive, Suit 200, Boulder Colorado, USA. *WiTilt v3*.

[76] Daniel Spelmezan, Mareike Jacobs, Anke Hilgers, and Jan Borchers. Tactile motion instructions for physical activities. In *Proceedings of the 27th international conference on Human factors in computing systems*, CHI '09, pages 2243–2252, New York, NY, USA, 2009. ACM.

[77] Stdint definitions, 2010. `http://blosc.pytables.org/trac/log/trunk/blosc/stdint-windows.h?rev=240`.

[78] H Stolze, J P Kuhtz-Buschbeck, H Drcke, K Jhnk, M Illert, and G Deuschl. Comparative analysis of the gait disorder of normal pressure hydrocephalus and parkinson's disease. *Journal of Neurology, Neurosurgery & Psychiatry*, 70(3):289–297, 2001.

[79] Texas Instruments Inc., Texas Instruments Post Office Box 655303, Dallas, Texas 75265. *ULN2803A Darlington Transistor Array*, Aug 2004.

[80] Vicon|applications|life sciences|sports|performance, 2010. `http://www.vicon.com/applications/sports_more.html#bibliography`.

[81] Daniel Vlasic, Rolf Adelsberger, Giovanni Vannucci, John Barnwell, Markus Gross, Wojciech Matusik, and Jovan Popovic. Practical motion capture in everyday surroundings. In *ACM SIGGRAPH 2007*, volume 35, 2007.

[82] BT Volpe, HI Krebs, N Hogan, OTR L Edelstein, C Diels, and M Aisen. A novel approach to stroke rehabilitation: robot-aided sensorimotor stimulation. *Neurology*, 54(12):1938–1944, May 2000.

[83] Wii at nintendo, 2010. `http://www.nintendo.com/wii`.

[84] Shandong Wu, Y. F. Li, and Jianwei Zhang. A Hierarchical Motion Trajectory Signature Descriptor. In *IEEE International Conference on Robotics and Automation*, pages 3070–3075, 2008.

[85] Xbee arduino library by andrew rapp, 2010. `http://code.google.com/p/xbee-arduino/`.

[86] Ungyeon Yang and Gerard Jounghyun Kim. Implementation and evaluation of "just follow me": an immersive, VR-based, motion-training system. *Presence: Teleoperators and Virtual Environments*, 11(3):304–323, 2002.

[87] KangKang Yin and Dinesh K. Pai. Footsee: an interactive animation system. In *2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 329–338, 2003.

[88] Yasuyoshi Yokokohji, Ralph L. Hollis, and Takeo Kanade. WYSIWYF Display: A Visual/Haptic Interface to Virtual Environment. *Presence: Teleoperators and Virtual Environments*, 8(4):412–434, 1999.

[89] Xiaoping Yun, E.R. Bachmann, H. Moore, and J. Calusdian. Self-contained position tracking of human movement using small inertial/magnetic sensor modules. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 2526–2533, April 2007.

[90] Ying Zheng and J.B. Morrell. A vibrotactile feedback approach to posture guidance. In *Haptics Symposium, 2010 IEEE*, pages 351 –358, Mar 2010.