Physically Animated Desktop Computer for Ergonomic & Affective Movement

by

Andrew Wang

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2006

Author .....
Department of Electrical Engineering and Computer Science
May 26, 2006

Certified by .....
Dr. Cynthia Breazeal
Associate Professor of Media Arts and Sciences
Supervisor

Accepted by .....
. Smith
Chairman, Department Committee on Graduate Students

Physically Animated Desktop Computer for Ergonomic & Affective Movement

by

Andrew Wang

Submitted to the Department of Electrical Engineering and Computer Science
on May 26, 2006, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

For better or worse, more people are spending their days in front of computers. With this increase in computer use, more people are complaining of back and neck pain. The simple act of changing your posture, however, can dramatically reduce the risk of back or neck injury from prolonged computer use. RoCo, the robotic computer, can encourage computer users to change posture by moving its screen to different positions. Having introduced motion, RoCo can now also begin to build a social relationship with the user and affect his affective state. This thesis describes the workings of RoCo and the results of an initial user study to produce affective movement.

Thesis Supervisor: Dr. Cynthia Breazeal
Title: Associate Professor of Media Arts and Sciences

Acknowledgments

Almost needless to say, thanks to my thesis advisor, Professor Cynthia Breazeal, for taking me on as a ninth graduate student and still finding the time to guide me.

Thanks to the entire Robotic Life Group. In particular, thanks to Andrea who helped me get started in the group as a UROP and provided support throughout the year, and thanks to Matt and Jesse who acted as my c5m technical support team. To Polly for helping me deal with the "real world" when it became neccessary.

Thanks to nearly everyone I know at MIT as they all came out to participate in my user study despite being as busy if not busier than me.

Thanks to John McBean for building an awesome robot like RoCo and for answering all my questions.

Thanks to Jasper and Jen who tolerated my frequent disappearences while working on our 6.883 final project.

Thanks to Wendi for helping make the final week of revising tolerable, if not fun at times, despite having three finals to study for herself.

And to all of my friends, who there are too many more to name individually, for always being there to help me get through those unmotivated times, for taking me out and showing me a great time.

To my sister, Eileen, who was always available and willing to chat over IM all the way from Australia.

To my parents for their love and support and doing all the right parent things both now and always.

Contents

1 Introduction **15**

1.1 System Overview 16

1.1.1 Physical Robot 17

1.1.2 Behavior System 17

1.1.3 Motor Control System 18

1.1.4 Sensory Input System 19

1.1.5 Intra Robot Communication System 19

1.2 Experiment Framework 19

1.3 User Study 20

2 Motor Control System **21**

2.1 Hardware Layer 21

2.2 Software Layer 22

2.2.1 Motion Programmer's Interface 22

2.2.2 meixmp_motor 24

2.2.3 meixmp 25

2.2.4 IRCP 26

2.3 Safety, Configuration, and Calibration 26

2.3.1 Configuration 26

2.3.2 Limits 26

2.3.3 Calibration 28

3 Sensory Input System **29**

3.1 Blue Eyes Pupil Tracking 29

3.2 IRCP 30

3.3 Future Systems 30

4 Experiment Software **33**

4.1 ExperimentModel 33

4.2 ExperimentView 34

4.3 ExperimentController 34

4.4 Results 35

4.5 Experiment Launcher 35

4.6 Implemented Experiments 36

4.6.1 Remote Associates Test Implementation 36

4.6.2 Tangram Puzzle Implementation 36

4.6.3 Tracing Puzzle Implementation 37

5 User Study **39**

5.1 Hypothesis and Predictions 40

5.2 Experiment Method 40

5.2.1 Subjects 40

5.2.2 Preliminaries 40

5.2.3 Success-Failure Manipulation 41

5.2.4 Posture Manipulation 42

5.2.5 Dependent Measures 43

5.2.6 Debriefing Procedures 43

5.3 Results 44

5.4 Discussion 46

6 Conclusion **49**

6.1 A New Research Platform 49

6.2 Future Work 50

6.3 Follow Up Studies 50

A Trace Puzzles **53**

B Tangram Puzzles **57**

C Remote Associates Test **61**

D Configuration **63**

D.1 Motor configuration 63

D.2 Experiment Launcher Configuration 69

E RoCo Wiring Setup **71**

E.1 Encoders 71

E.2 Limit Switches 71

E.3 Motor card Output 72

E.4 Servoamplifier Setup 72

F Experiment Data **75**

List of Figures

1-1 RoCo: A Robotic Computer 17

2-1 Main methods of `meixmp` module 26

2-2 Partial `meixmp` configuration file 27

5-1 Two soluble and two insoluble puzzles 41

5-2 Results chart 42

5-3 RoCo's postures: slumped, neutral, and upright 42

A-1 Success Outcome: All solvable 54

A-2 Failure Outcome: Unsolvable, solvable, solvable, unsolvable 55

A-3 Persistence Task: All unsolvable 56

B-1 Tangram puzzles: First four 58

B-2 Tangram puzzles: Last three 59

C-1 Remote Associates Test 62

E-1 A diagram of the PCI control card 72

E-2 Pinout diagram for axis 0 73

E-3 Single-ended encoder wiring 74

F-1 Success-Neutral Data 76

F-2 Success-Slumped Data 77

F-3 Success-Upright Data 78

F-4 Failure-Neutral Data 79

F-5 Failure-Slumped Data 80
F-6 Failure-Upright Data 81
F-7 Success-Neutral Data 82

List of Tables

4.1 `Results` Interface 35

5.1 Average number of tracing attempts 44
5.2 Average number correct on Remote Associates Test 45
5.3 Average number of Tangrams solved in 7 minutes 45
5.4 Average time to solve first Tangram is seconds 45
5.5 Average self-reported comfort levels 48

D.1 Motor Configuration variables 68

F.1 Page letter and corresponding test ordering 75

Chapter 1

Introduction

An ever increasing number of Americans, approximately 75%, spend the bulk of their work day in static sitting postures, often in front of computers. An ever increasing number of Americans also suffer from musculo-skeletal problems. In particular, reports of lower back pain and discomfort have risen [6]. The correlation between these two trends is not coincidental. Consequently, the term ergonomic has become a major selling point for chairs, tables, and the like. Studies have shown, however, that physical movement is one of the simplest and most effective preventive measures for back pain [13]. Individuals who change their posture throughout the day can better sit for prolonged periods. Furthermore, continuous movement is not only therapeutic for joints and ligaments, but the associated muscle movement also improves circulation. In current workplaces the burden is on the individual to change his posture throughout the day. One solution to this problem would be to have a computer program remind the user to take a break. Indeed such programs already exist to help prevent RSI and other typing related injuries. But, the burden still rests on the user's shoulders to heed the program's advice. With social robotics and a slight but important paradigm shift, we can do better.

Robots and computers are traditionally seen as passive entities designed to serve their human operators and to make life easier. Autonomous robots exist, but only so they can operate without constant human supervison, such as assembly line robots. In most common human robot interactions the robot is a passive agent. By allowing

our robots to take on a more active role, we also allow them to better assist us with our tasks. To explore this hypothesis, we have developed RoCo, a physically animated desktop computer.

When people collaborate, they tend to move in a variety of reciprocal ways. Research has shown that people will also frequently mirror the posture of a robot when engaged in a social interaction, much as they do with other people [2]. As a social robot, RoCo can encourage this mirroring behavior as a method to adjust the user's posture. RoCo can also use the tactic of moving to a position such that the user has to adjust to better view the screen. With these behaviors, RoCo takes a more active role to subtly and unobtrusively induce ergonomic movement.

The benefits of RoCo's movement extend beyond the realm of ergonomics. As discussed previously, when people collaborate, they engage in a variety of reciprocal movements. These movements not only serve as vital nonnverbal cues, but they also act to build social rapport. So called "immediacy behaviors" such as forward leaning, nodding, frequent gesturing, and postural openess all project liking and engagement [1], [14]. Christensen & Menzel [4] also showed that "immediacy behaviors" in teachers also increase learning outcomes. And in general, collaborations between friends is a more effective learning expeirence than collabrating with acquaintences [10]. Finally, even in the abscence of social interaction, posture combined with affective state can also affect such things as persistence and feelings of control. By inducing the appropriate posture following a success or failure, RoCo can help maximize or minimize the effects of success or failure respectively [15]. Hence, RoCo can alter not only the user's physical posture, but his cognitive and affective state as well.

1.1 System Overview

RoCo is composed of four sub-components: the physical robot itself, the behavior engine, the motor control system, and the sensory input system.

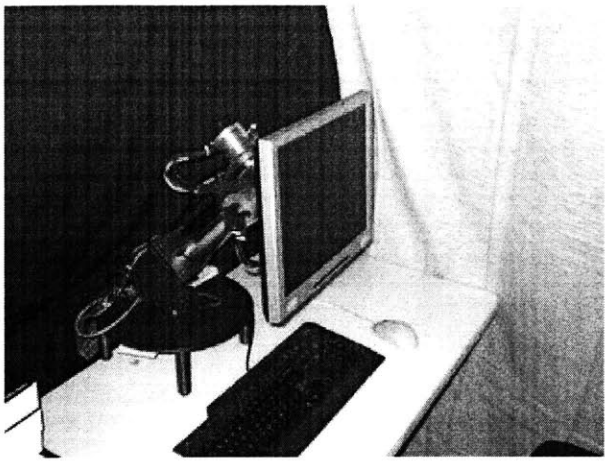

Figure 1-1: RoCo: A Robotic Computer

1.1.1 Physical Robot

The physical robot has five degrees of freedom that manipulate a mechanical neck with a LCD screen, it's "head", mounted on it. See figure 1-1. The robot has no explicit facial features although the LCD screen could display them graphically if necessary. The LCD screen's primary task, however, is to display task relevant information like any other computer monitor.

Three degrees of freedom control head motion: head yaw, head pitch, and head roll. Two additional degrees of freedom control the neck: base yaw and base pitch. These five degrees allow RoCo to perform a wide variety of simple motions, including nodding, shaking its head, and leaning forward. These life-like motions are sufficient to implement a wide variety of the "immediacy behaviors" and postural changes.

1.1.2 Behavior System

The behavior system is built on the c5m architecture. The c5m system was originally developed to create autonomous, animated, interactive virtual characters. The Robotic Life group at the MIT Media Lab has adapted it to control and simulate physically animated robots. The details of c5m are far beyond the scope of this thesis. They can be found in [3]. The following is a brief high level description of the barebones setup required to use RoCo in a simple experiment.

Fundamentally, the c5m architecture consists of a Perception System, a Belief System, an Action System, and a Motor System. The Perception System receives various data from external sensors (see the description of the Sensory Input System). It processes that data into a consistent internal format that the other systems can operate on. It then sends the data to the Belief System.

The Belief System maintains persistent knowledge about the world. It uses the processed sensory data to update and create knowledge. Stale knowledge eventually gets removed as part of a culling process. The Action System examines the "beliefs" of the system and determines the next action the agent should take. This process is called triggering. Triggered actions issue the appropriate commands to the Motor System.

The Motor System, perhaps more appropriately named the Motor Render System, plays and blends animations for the simulation and also broadcasts joint angle positions to the motor control system. An operator can also possible to manually control the joint positions through a graphical user interface, bypassing the perception, belief, action interaction. This mode of operation is useful for debugging and testing. The Motor System issuses its commands over IRCP to the Motor Control System which actually drives the physical motors. IRCP and the Motor Control System are overviewed below, and the Motor Control System is detailed in chapter 2 as well.

1.1.3 Motor Control System

The Motor Control System is built on top of the Motion Programmer's Interface (MPI) API from the Motion Engingeering Inc (MEI). It is extension of the motor control system used to drive Leonardo's motors. It controls the actual movement of the motors through an MEI motion control card. It receives joint angle positions from the behavior system that it translates into encoder position counts. Those counts are used to drive the physical motors.

Additionally, the motor control system is fully configurable with a plain text configuration file. Based on values in that configuration file, the motor control system is capable of auto calibrating the joints on startup. It also implements both hardware

and software based limit switches to prevent RoCo from damaging itself and other objects that might be near it.

1.1.4 Sensory Input System

Finally, the Sensory Input system consists of a number of independent input devices that presently include the computer's mouse and keyboard, and a blue eyes pupil tracking system. Future sensory devices will include a head pose tracker and a sensor chair. These sensors tell RoCo about the world. They enable it to model and monitor the user's emotions and attention. From that information, RoCo can determine appropriate times to change its posture and what optimal posture to adopt.

1.1.5 Intra Robot Communication System

With the exception of Motor Control to Robot communication which is done through the MEI control card, all intra-system communication is done over a local area network using the intra-robot communications protocol (IRCP), a custom UDP based protocol developed by the Robotic Life Group [8]. The heart of IRCP is the packet data structure. Each packet contains a `robotID`, along with source and destination module ids. By convention, each robot on the network claims a unique id. Similarly, each module or system within the robot is claims a unique module id. In principle then, a module can simply transmit its packet over the broadcast address and only the targeted robot and module will accept it. In practice, auto module address discovery can be used to reduce network traffic.

1.2 Experiment Framework

I also developed an experiment framework to facilitate human subject studies. The framework consists of a number of abstract class and interfaces to simplify the creation of new experiments. The framework is based on the model-view-control design pattern and designed with integration with IRCP packets in mind.

For the initial study, six experiments have been created. A detailed overview of the framework and implemented experiments follows in Chapter 4.

1.3 User Study

To evaluate RoCo's effectiveness in encouraging healthy and affectively beneficial movement, I have duplicated Riskind's "stoop to conquer" experiment [15]. In his original study, Riskind showed that a stooped or slumped posture helped people better cope with failure while upright postures helped people better exploit success. When people adopted incongruous postures (stooped/slumped upon success, upright upon failure), they felt like they had less control, showed less motivation in persistence tasks, and higher depression than subjects in congruous positions.

In the original experiment, the subjects were asked to either slump or sit upright under the false pretense of a biofeedback experiment. While this kind of manipulation is uselful and important for detecting the "stoop to conquer" effect, it is impractical for taking advantage of it. By changing RoCo's posture, we can instead subtly lure the user into the optimal position without seriously interrupting his workflow. The study is designed as an initial baseline study that investigates RoCo's ability to manipulate both the user's posture and the user's cognitive and affective state. The results of this study provide a useful benchmark to judge future work against. It also serves as a practical test of the RoCo system itself.

Chapter 2

Motor Control System

The Motor Control System is comprised of the physical hardware and the software that controls it. The hardware can be broken into two components: RoCo the robot (motors, amplifiers, limit switches) and the control hardware (encoders, motor control board, control PC). The software layer consists of the Motion Programmer's Interface and a higher level driver that integrates the motor system with the c5m architecture. The top layer of abstraction in the motor control system is the safety and configuration layer. Behavior system developers should concern themselves primarily with this layer.

2.1 Hardware Layer

The physical robot was designed and built by Xitome Inc. It has five degrees of freedom: head yaw, head pitch, head roll, base pitch, and base yaw. The "head" refers to a mounted LCD screen that displays relevant task information. The motors are brushless DC motors that operate smoothly and quietly to avoid distracting the user.

An eight axis analog XMP control card from Motion Engineering Inc. (MEI) is used to interface with the control PC and the robot's servoamplifiers. The servoamplifiers are configured for current control mode (i.e. torque control). Encoder feedback goes to the MEI card which handles the position control via software running on the

control PC. Because the MEI card expects differential encoders and RoCo has single ended encoders, there is also a bias circuit to provide an appropriate offset as specified in the MEI user manual. See the appendix for detailed wiring diagrams.

2.2 Software Layer

The motor control software layer builds upon the existing systems in the Leonardo platform. The details of the Leonardo motor system can be found in chapter 5 of Matt Hancher's Master's thesis [8]. Briefly, the motor system is designed with abstractions to seperate the programmer and the engineer.

To support RoCo's hardware using this motor system library, two new classes have been created: `meixmp_motor` and `meixmp` that derive from `motor` and `abstract_motor_container` respectively. These classes interface with the hardware API, also known as the Motion Programmer's Interface.

2.2.1 Motion Programmer's Interface

The Motion Programmer's Interface (MPI) is the API for interfacing with the MEI XMP control card. It is, therefore, the basis for the implementations of `meixmp_motor` and `meixmp`. MPI is a very flexible, low level API desgined to handle nearly every motor system imaginable. It is object oriented and written in C.

At its core, MPI and the XMP software architecture is composed of four separate modular objects: Motors, Filters, Axes, and Motion Supervisors. The Motion Supervisor can control one or more Axes of motion. It is the primary interface for handling motion. It also monitors the status of all the axes under its control.

An Axis represents a single physical axis or degree of freedom. Given trajectory information from the supervisor, the Axis object genereates the desired path. Each Axis in turn can map to one or more Filter.

A Filter is essentially the feedback control object. It computes the output (usually a voltage level) based on data (usually command positions) from the Axis object.

Finally, a Filter maps to a Motor object. Motor objects represent the physical

motors themselves and in this architecture behave essentially as I/O objects. They provide input data about the physical motor, such as encoder position counts, the state of limit switiches, and status signals. They provide output directly to the physical motors as commanded by the associated filters and axes.

Additional MPI objects that RoCo's motor system employs are the Event, Notify, and the Event Manager objects. Event objects are accesible through the Event Manager and provide information about an asynchronous event. That information minimally includes the event type and source. Notify objects are used to listen for these events and trigger event handling mechanisms. The Event Manager, as the name implies, manages the interaction of Event and Notify objects. It obtains asynchronous events, generates appropriate Event objects, and dispatches to registered Notify objects.

Finally, there is the Control object which represents the control board itself. Every application creates a single Control object per board. A Control object is required to create the other objects.

A simple single axis move, thus, requires a fair amount of code, part of which is shown below:

```
/* Create motion controller object */
*control = mpiControlCreate(controlType, controlAddress);
msgCHECK(mpiControlValidate(*control));

returnValue =  mpiControlInit(*control);
msgCHECK(returnValue);

/* Create motion controller object */
*control = mpiControlCreate(controlType,
                            controlAddress);
msgCHECK(mpiControlValidate(*control));

/* Initialize motion controller */
returnValue = mpiControlInit(*control);
msgCHECK(returnValue);

/* Create axis object */
*axis = mpiAxisCreate(*control, axisNumber);
msgCHECK(mpiAxisValidate(*axis));
```

```
/* Create motion supervisor object with axis */
*motion = mpiMotionCreate(*control,
                          motionNumber,       /* motion supervisor number */
                          *axis);             /* axis object handle */
msgCHECK(mpiMotionValidate(*motion));

MPIMotionParams     params;         /* Motion parameters */

returnValue = mpiMotionStart(motion,
                             MPIMotionTypeTRAPEZOIDAL,
                             &params);
msgCHECK(returnValue);
```

The `meixmp_motor` and `meixmp` classes abstract this complexity away and provides high level behavior programmers with a simpler and cleaner interface.

2.2.2 meixmp_motor

The `meixmp_motor` object represents a single one to one mapping configuration of MPI objects: Motion Supervisor to Axis to Filter to Motor. The `meixmp_motor` class, thus, only represents a subset of configurations that MPI can handle. In this instance we give up the full flexbility that MPI offers for a simpler, cleaner, and more robust implementation. Future work may wish to extend `meixmp_motor` to handle additional configurations or create separate motor classes for them.

The `meixmp_motor` handles the creation and deletion of the low level MPI objects. Because it inherits from the `motor` object, it also provides a number of high level commands like `set_target_position` and `get_velocity`. Using the `meixmp_motor` object, a single axis move becomes much easier. The same single axis move shown in the previous section now looks something like:

```
// create the motor system using the specified config file
// this creates all the mappings etc... from previous example
motor_system main_roco(config_file);

// set the target positions to perform the move
for(motor_system::iterator i=roco->begin(); i!=roco->end(); ++i) {
```

```
    i->set_target_position(position);
}
```

2.2.3 meixmp

The `meixmp_motor` object is a low level driver for controlling a single axis. To control a collection of motors the system also provides an `meixmp` object. It extends the `abstract_motor_container` class from Leonardo's system and represents a mid-level driver for controlling multiple motors. It spawns a separate thread that runs the actual control code and communicates with the hardware.

The `meixmp` module wraps the MPI Control object. The `meixmp` module is responsible for creating, destroying, and maintaining the individual `meixmp_motors` in the system. This module also creates and manages an Event Manager object, which as described previously, generates event objects for enabled event sources.

When the `meixmp` module starts up, it loads the appropriate configuration file and creates all the `meixmp_motor` objects as specified. Once all of the `meixmp_motor` objects have been created, the module enters its auto calibration phase where it discovers the limits of each motor and moves them to their origins. The details of this auto calibration phase are described in the next section.

After auto calibration, the module spawns a main thread of execution. In the main thread, the `meixmp` module continually polls the event manager and motors for critical error events such as limit and position errors. It also tries to move the motors to their target positions. Actual movement commands for the motors, on the high level, are sent by the behavior system via IRCP. On the low level they are handled by callbacks. The main thread also waits for a termination signal after which it disables all of the motors it controls and terminates.

A listing of the more important methods in `meixmp` is shown in Figure 2-1.

Function	Description
`void enable()`	Enable the joint
`void disable()`	Disable the joint
`void set_target_position(float)`	Set the joint's target position
`void poll()`	Polls the event manager for events

Figure 2-1: Main methods of `meixmp` module

2.2.4 IRCP

The IRCP setup for RoCo is nearly identical to the setup for Leonardo. The details for the IRCP protocol itself can be found in Hancher [8]. RoCo's motor system supports the same set of low-level motion commands as Leonardo as defined by the IRCP major type 0. These commands are "Request Response", "Enbable Motors", "Disable Motors" and "Set Target Positions".

2.3 Safety, Configuration, and Calibration

2.3.1 Configuration

At the highest level of abstraction, `meixmp_motor` and `meixmp` consist of a single configuration file that can be edited by any engineer familiar with motor control systems to tweak the performance without recompiling the code. The entire system appears as a collection of familiar variables like `velocity` and PID coefficients. This conveniece arose directly from integrating with Leonardo's existing motor system. Part of the configuration file used for RoCo is shown in Figure 2-2. The meanings of each variable are included in the appendix.

2.3.2 Limits

RoCo is also configured with a number of hardware and software safety features that prevent it from damaging itself and other things around it. Hard stops prevent each joint from rotating too far, and limit switches notify the software when a stop has been hit. By default, when RoCo reaches a hard stop, the motor is immediately

```
name = "RocoMotorSystem"
description = "RoCo Motor System"

MEIXMP {
     type = "MEIXMP"
     description = "RoCo Controller"

     HeadYaw {
          name = "headRoll"
          description = "head yaw"
          motion_number = 0
          axis_number = 0
          filter_number = 0
          motor_number = 0
          motor_type = 0
          amp_polarity = 1
          pgain = 75
          igain = 2
          dgain = 55
          imax_moving = 0
          imax_idle = 10000
          drate = 7
          range = 43000
          invert_angles = 1
          min_angle = -0.523
          max_angle = 0.523
          pos_hw_action = 3
          pos_hw_polarity = 0
          pos_hw_direction = 0
          pos_hw_duration = 0.1
          neg_hw_action = 3
          neg_hw_polarity = 0
          neg_hw_direction = 0
          neg_hw_duration = 0.1
          velocity = 3000
          acceleration = 5000
          deceleration = 5000
     }
}
```

Figure 2-2: Partial `meixmp` configuration file

disabled to prevent it from trying to push through and damaging itself [1]

Software limits can also be set which trigger when the software detects the encoder position has reached a certain point. By default, triggering this event will also disable the motor. This action can be changed through the configuration file.

The third safe guard is a position error limit. This is another software safety measure that triggers if the commanded position and the actual position differ more than some specified amount. This measure is used to prevent the motors from trying to push through objects that RoCo might collide with that are in its range of motion.

2.3.3 Calibration

On startup, RoCo will try to auto-calibrate itself using the information provided in the configuration files. For each joint, RoCo will move until a hard stop is detected. The encoder count at this point is used as a reference point and mapped to the max angle defined in the configuration file. The range, also read from the config file, is used to determine the encoder position of the min angle. The joint then moves to the origin, the 0 angle.

If the motor system detects an unexpected error state or starts up in an error state, the auto-calibration sequence will not complete. Instead, the motor system will report the error, disable all motors, and exit. As described, the auto-calibration phase relies on several values in the configuration files that must be predetermined by an engineer. The easiest way of determing encoder ranges is to use the Motion Console utility provided by MEI. The Motion Console is essentially a GUI for MPI designed for testing and monitoring motion control components.

[1] For some motors, disabling them means letting gravity take over and can cause the joint to fall. The base pitch joint is particularly prone to this.

Chapter 3

Sensory Input System

RoCo is designed to support a wide variety of sensor and input systems that will inform it about the world in addition to the standard mouse and keyboard. As with the other system modules, each sensory and input device interfaces with the behavior system using IRCP.

3.1 Blue Eyes Pupil Tracking

The first external sensory system that RoCo supports is the Blue Eyes pupil tracking camera built by IBM. The Blue Eyes camera uses a combination of off-axis and on-axis infrared LEDs and camera to track the user's pupils in real time by producing the red eye effect [9]. Additional real time techniques developed by Kapoor and Picard [12] automatically detect and track other user facial features such as eyes, brows, nose and lips. There is also support for blink and nod detection. This information will be important for determining the user's attention which will serve as a cue for appropriate times to shift RoCo's posture.

The blue eyes system is mounted on the bottom of RoCo's monitor and captures images in real-time for detection. The pupil locations are determined and sent to RoCo over IRCP. To improve robustness and reduce variability in reported pupil position, the blue eyes module does some smoothing and filtering. For the last nine points, it does a vote of valid and invalid pupil points (invalid points have negative

values). Each vote is weighted linearly according to recentness with the most recent points receiving the most weight. If the system determines that it is indeed detecting pupils, it takes a weighted average of the valid points, again weighted by recentness. This averaged position is sent to the behavior system. The smoothing process introduces a slight delay in tracking of the pupils in return for less variablility in the position.

3.2 IRCP

The Blue Eyes System sends left and right pupil positions as a float array fo length 4 over IRCP to the behavior system. The array contains in this order: left pupil x position, right pupil x position, left pupil y position, and right pupil y position.

The Blue Eyes System introduces a new major type to IRCP, type 6, which is the first unreserved type. Any IRCP compatible systems that are interested in Blue Eyes data should subscribe to major type 6. The pupil information is sent using minor type 0. Additional minor types can be defined for additional data, such as pupil size, eye points, eyebrow points, and nod and blink detector. Currently, RoCo's behavior system implements a simple Blue Eyes packet handler which simply receieves the pupil data and prints it out.

3.3 Future Systems

Future sensory systems will include a head pose tracker as well as a sensor chair. The head pose tracker will be used to track the orientation of the user's face, information that is difficult to obtain from the Blue Eyes system. When used in combination with the Blue Eyes camera, the head pose tracker will provide additional cues for the user's attention and focus.

Similarly, the sensor chair will not only provide posture data for healthful movement applications, but it can also provide information on affective states such as high and low interest levels or even pride and disappointment. Changes in posture will

also signal appropriate times for RoCo to change its posture as well.

Chapter 4

Experiment Software

Running a user study on the RoCo platform required the development of a flexible experiment framework that allows simple experiments to be readily created and deployed. The experiment framework is platform independent, and it is designed to easily interface with the c5m architecture via IRCP. In this context, an experiment refers to a user experiment, one that has a human subject as the primary agent.

The framework follows the Model-View-Controller (MVC) design pattern as a typical experiment lends itself nicely to this approach. The model encapsulates the abstraction of the experiment, while the view provides the user interface, and the controller manages that interaction by calling methods between the view and the model.

The following sections detail the MVC components and the provides an overview of currently implemented experiments.

4.1 ExperimentModel

In a typical experiment, the user is presented first with the instructions, then the experiment task, and finally the results. The `ExperimentModel` supplies all of the experiment logic and data, including instructions and results. For example, for the Remote Associates Test, the experiment model provides a list a triples, knows the correct answer for the triple, and also stores the subject's answer.

The `ExperimentModel` interface specifies a `getResults()` and a `getInstructions()` method. Each individual implementation is free to provide its own data and procedure specific methods. The `ExperimentModel` provides a layer between the experiment relevant data and the user interafce used by the subject to complete the experiment. This design enables the experiment developer to freely change internal representation and add additional logic without changing the user interface.

4.2 ExperimentView

From a high level perspective, the `ExperimentView` handles the display of the experiment instrutions, tasks, and results, all of which are supplied by the model. The view also publishes user interface events such as mouse clicks and button clicks that other objects, like the controller, can subscribe to. In the current implementation, all views use Swing, the default Java GUI framework, but a developer can easily change them to use their GUI toolkit of choice. The separation into model and view allows this modification to happen without changing the underlying model.

The implemented `ExperimentView` abstract class extends `JPanel` provides a framework for creating similarly layed out experiment views. The main view area is called the *center*, and can be set to any `JComponent` object with a call to `setCenter()`. Thus, a standard method of creating new experiment views is to extend `ExperimentView` and create an experiment specific pane that gets set as the center on a call to `showExperiment()`. A `showPage()` method also provides a convienent way of displaying html pages in the *center*, which can be used for showing instructions or results.

4.3 ExperimentController

The `ExperimentController` handles the interaction between model and the view. It subscribes to the events published by the view and interprets them by calling the appropriate methods in the model. In short, the controller is a glorified event

Method	Description
`String[] asStringArray()`	Return the results as a string array
`Integer[] asIntegerArray()`	Return the results as an integer array
`Float[] asFloatArray()`	Return the results as a float array
`String asFileString()`	Return the results as string to be written to file

Table 4.1: `Results` Interface

handler. The `ExperimentController` also publishes an event, *results available*. The *results available* event signals that the current experiment has ended and a `Results` object is available. Two common subscribers to this event is an IRCP module and the experiment launcher. The IRCP module can take the results and send them to any listening robots like RoCo, while the experiment launcher saves the results to file.

4.4 Results

`Results` objects get created by the `ExperimentModel` and are accesible via the `getResults()` method. They are also available as a parameter of the *results available* event. The `Results` object provides a number of views for the underlying data. Views include arrays of numbers, strings, or just a single string. Using the appropriate view makes it easy to send the data over the network with IRCP or write it to a file. Table 4.1 lists the `Results` interface methods.

4.5 Experiment Launcher

The `Launcher` is an application that allows an experimenter to select and execute an experiment configuration. An experiment configuration is a series of `Experiments` that the `Launcher` will run in sequence. The `Launcher` collects the results from each experiment (it implements the `ResultsListener` interface) and writes them to a file. The `Launcher` application also adheres to the MVC model in the same way the `Experiment` classes do.

Experiment configurations are defined in an xml file specified to the `Launcher` at runtime. The xml file lists which `ExperimentController` along with construc-

tor arguments. The Launcher uses Java's reflection API to run each experiment in sequence. See the appendix for a sample file.

4.6 Implemented Experiments

I implemented six full experiments for the user study. Three of them are very simple. The demographic experiment simple presents the user with a form that collects demographic information. The Quicktime experiment shows the user a Quicktime video. The debrief experiment, like the demographic experiment, presents a form for the user to complete. The other three experiments are modeled after standard tasks used in a variety of psychology experiments.

4.6.1 Remote Associates Test Implementation

The Remote Associates Test experiment runs the Remote Associates Test by showing each triple individually. The triples and their answers are specified in a file. The implementation of the Remote Associates Test is very straightforward. It presents the three words and a text entry box to receive the user's answer. Results are stored as "triple, guess, answer, correctness", e.g. "shelf, end, read, book, book, true" and "elephant, vivid, lapse, wrong, memory, false". Both the guess and answer are stored so that a human scorer can review the results as the computer is unable to detect things such as spelling errors and obvious typos.

4.6.2 Tangram Puzzle Implementation

The Tangram Puzzle experiment presents the user with a series of tangram puzzles to solve. The tangrams are a predefined set of seven pieces, but the puzzles are specified in files. The user manipulates the tangram pieces by mouse to move, rotate, and flip them. Standard geometry algorithms for convex shapes are used to perform these tasks. An important note is that these functions assume that all polygons have the same handedness, (i.e. vertices defined in same direction, either clockwise or

counterclockwise). Inconsistency in this aspect will lead to inaccurate results.

Solution checking is done by computing overlap. A puzzle is considered solved if, within some tolerance level, the puzzle has been covered and none of the pieces overlap. The tolerance level was determined through user testing and for the study was set to 7%. This means that 7% of a piece could lie outside the puzzle. It also means that 7% of a piece can overlap with another piece. Although this scheme allows for certain non-solutions to pass, user studies have shown that the user will not attempt non-solutions to the point where triggering the solved condition is likely.

4.6.3 Tracing Puzzle Implementation

The tracing puzzle experiment presents the user with a series of line drawings that he must attempt to trace without retracing any lines or lifting up the pen. The tracing process can either be simulated with a mouse button hold and drag, or more intuitively, with a stylus pen and tablet.

Trace puzzles are stored internally as an undirected graph of edges and vertices. A solution for the tracing puzzle is one where every edge has been "marked" exactly once. There are two kinds of puzzles: solvable and unsolvable. Solvable puzzles have at most two odd degree vertices. Unsolvable puzzles have more than two. Without delving into a rigorous mathematical proof, the intuition is that every vertex must have an even degree because if you enter you must be able to leave via a different edge. The two exceptions to that rule are the first and last vertex, which is why there can be up to two odd degree vertices. The implementation first checks the solvability of a puzzle based on this criteria. If it's unsolvable, it performs no further checks and always returns false when the `isSolved()` method is called.

For the solvable case, the program attempts to map the user's drawing to the graph. The process works as follows: Start with first point that the user drew and iterate until the point is within some radius of a vertex. That vertex, P, is the labelled the "start vertex" and all points before the "start vertex" are moved to end of the list.

The next step is to determine the next vertex the user is attempting to reach. All

points within a certain radius of P are ignored as during this period the user is often deciding where to go next and the point data are unreliable in predicting the path. The first ten points outside the cuttoff radius are averaged, call it Q, and the line segment PQ is used to select the best matching edge. The simplest way of selecting the best edge is to take the normalized edge vector (PR) that maximizes the dot product with PQ.

The next step is to walk the edge to the next vertex, R. For each user traced point, the program checks that it's within a certain distance of the selected edge until the point falls within the cutoff radius of R and the process of finding the next vertex begins again, where R becomes the new "start vertex". If an edge PR is successfully "walked" in this manner, it is considered "marked". If the points are ever too far from the selected edge or if R is never reached, the puzzle is considered unsolved.

Similarly, the program tracks each edge as it is "walked" and if an edge is ever "walked" more than once or not at all, the puzzle is also considered unsolved. If every edge has been "walked" exactly once, the puzzle is considered solved.

Like the Tangram Puzzle, the introduction of techniques to compensate for human error introduces the possibility of missing some solutions. Again, the appropriate cutoff points were determined during a pilot study.

Chapter 5

User Study

The RoCo platform is a novel approach to the issue of human robot interaction. By taking the place of an everyday machine, the desktop computer, RoCo represents a significant intrusion into many people's daily lives and is at risk for becoming more distracting and annoying than it is useful. It is an open question whether reciprocal physical movement of human and computer can not only be designed to promote back health but also improve task efficacy. Earlier studies suggest that the answer to both questions is "yes". A series of Human Robot Interaction studies showed that people frequently mirror the posture of robot when engaged in a social interaction [2]. A number of psychology studies have also explored the effect of body posture on affect and cognition [16],[15],[5],[18].

The objective of this first study is to provide a proof of concept for RoCo both as a viable research platform and as a potential robotic companion. It serves as a baseline study for future studies with RoCo, and the act of conducting the study provides valuable feedback to improve the design of the platform.

This initial study expands on the results of Riskind's [15] "stoop to conquer" study. Briefly, the original study showed that physical posture, like facial expressions [5], can not only indicate mental state but may also affect the mental state and behavior. The results suggest that "incongruous" postures, such as slumping after a success, can negatively affect subsequent performance, while "congruous" postures, such as slumping after a failure, help to mitigate the effects of failing.

5.1 Hypothesis and Predictions

The experiment expands on the appropriateness hypothesis [16] which predicts that congruous posture guide an individual towards self-regulating behaviors while incongruous posture leads to self-defeating behavior. Taking advantage of the unique RoCo research platform, the experiment introduces a different posture manipulation method that allows the subject to perform dependent measure tasks while in the manipulated posture. The expectation is that RoCo will be an effective agent for manipulating posture and inducing the "stoop to conquer" effect.

5.2 Experiment Method

This experiment measures persistence on a helplessness task, creativity on a word association task, and general spatial cognition on a puzzle task as a function of congruous and incongruous postures following affect manipulation.

5.2.1 Subjects

71 naive subjects were gathered from MIT and surrounding area. Subjects were given a $10 gift certificate to Amazon.com as compensation for their participation in the study. Subjects were assigned to one of the six conditions based on the order that they signed up to participate in the study. The first subject was assigned to condition one, the second to condition two, etc. Hence, randomness was achieved through the signup process which was done through postering, mailing lists, and craigslist.

5.2.2 Preliminaries

When subjects arrived they were first greeted by the experimenter then led to a standard PC. The experimenter read the following standard set of instructions aloud to the subject:

"Please be seated. In front of you is a standard computer setup with mouse, keyboard, monitor and a pen tablet for use in the tracing puzzles. You may arrange

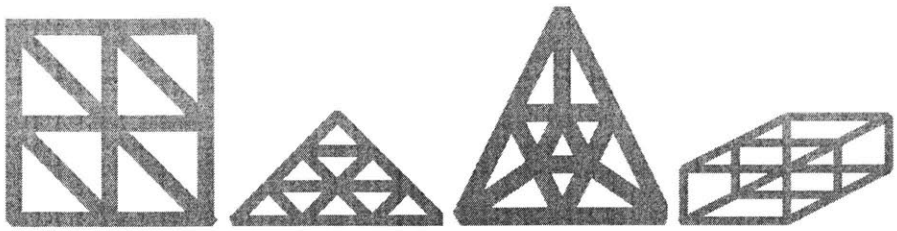

Figure 5-1: Two soluble and two insoluble puzzles

these components on the desk any way you like. Please read the instructions carefully as you go. The height of the chair is adjustable with a lever underneath the seat. I will be outside the curtains, if you have any questions or get confused, but in general, please try do as much on your own as possible."

The experimenter then left the area while the subject was shown a two minute video clip known to induce neutral affect [17].

5.2.3 Success-Failure Manipulation

Subjects were randomly assigned to either a success or failure condition according to the order they registered for the experiment. They were given a series of four tracing puzzles to solve. They had two minutes to solve each puzzle. To solve a puzzle, the subject must trace over the design without lifting a pen from the puzzle or retracing any lines. In this case, the puzzles were presented on a standard LCD screen and pen tracing is done with a computer pen and tablet input device. The puzzles used are the same set used by Riskind [15] in his studies as well as by Glass and Singer [7]. An example of the puzzles is shown in Figure 5-1. The complete set of puzzles can be found in the appendix.

To create a success condition, all four puzzles were solvable. Generally each subject was able to solve at least three out of the four. Unsolved puzzles were usually the result of not carefully reading the instructions beforehand or difficulty using the pen and tablet interface. Regardless of how the subject actually performed, a results chart, shown in figure 5-2, displayed and they were told that they scored an 8 out of 10.

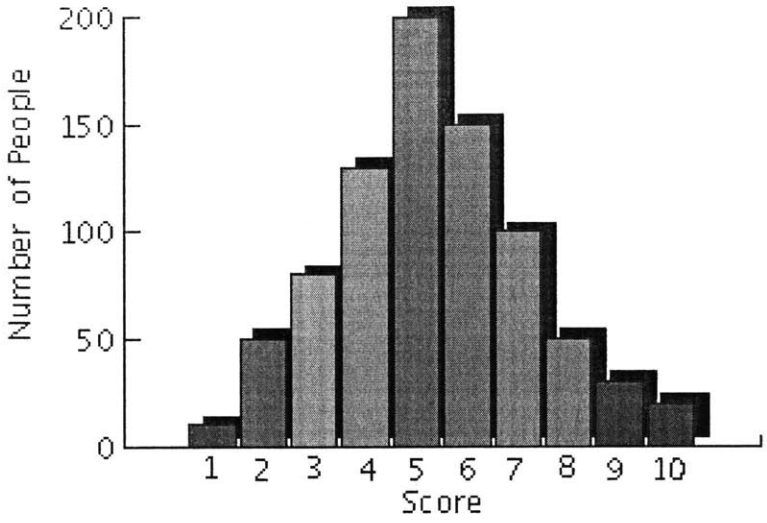

Figure 5-2: Results chart

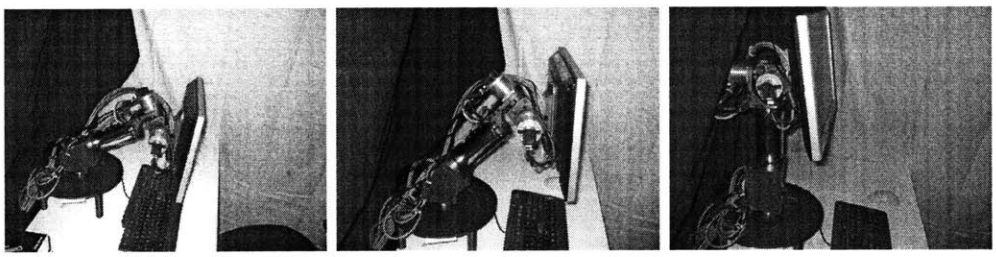

Figure 5-3: RoCo's postures: slumped, neutral, and upright

For the failure condition, the first and last puzzles were unsolvable. The sense of failure was further reinforced by displaying the same results chart as in the success condition except in this case they were given a score of 3 out of 10.

5.2.4 Posture Manipulation

Following the success-failure manipulation, the subject was taken to RoCo who's position had already been preset to either slumped, upright, or neutral. These positions are shown in Figure 5-3. The subject was seated in the same calibrated chair and asked to perform another series of puzzles, this time on RoCo. The subject was video taped from the side as a manipulation check.

5.2.5 Dependent Measures

The experiment examined three dependent measures: creativity, spatial cognition, and persistence. Subjects were administered these tests in a random order to minimize any effect the order of the tasks might have.

Remote Associates Test - The subject was asked to complete 14 items of the Remote Associates Test that ranged from easy to hard. Past research [11] has shown that performance on the Remote Associates Test improves with positive affect, although negative affect does not have an adverese effect. The expectation is that mismatched postures will dampen the impact of positive affect.

Unsolvable Tracing Task - The subject was also given a helplessness task to measure his persistence on an insoluble task. The subject was given four mathematically unsolvable tracing puzzles with a time limit of two minutes for each. This task assumes that the fewer number of tries in the allotted time, the lower the subject's tolerance for a frustrating task. Some of the puzzles are the same as those used in Riskind's original study. Additional puzzles were created by transforming some solvable into unsolvable. Debriefings showed that only people who knew the mathematical rule ahead of time for solvability were able to distinguish solvable from unsolvable puzzles.

Tangram Puzzle - The subject was given seven minutes to try and solve as many (up to seven) tangram puzzles as possible. Good performance on tangrams has been linked with good spatial cognition. A maximum of seven was chosen since even for someone who knew the solutions ahead of time could not complete all seven in seven minutes.

5.2.6 Debriefing Procedures

Following the dependent measure tests, the subject was given a full debriefing. As a check on the success-failure manipulation, the subject was asked how well they think they performed on the first part. All subjects in the failure manipulation responded

Outcome	Slumped	Neutral	Upright
n =	*10*	*11*	*9*
Success	8.15	8.32	11.97
n =	*12*	*10*	*11*
Failure	8.33	8.75	8.41

Table 5.1: Average number of tracing attempts

with answers like "not well", "below average", and "ok", suggesting that the manipulation was sucessful. Similarly, most subjects in the success case responded with answers such as "well" and "above average". Four subjects in the success condition who had trouble with the tracing puzzles in part one did report that they did not do well. Their results have been discounted.

Following the manipulation check, I disclosed the details of the study including the impossibility of some of the tracing puzzles and the fabricated test results in part one. Four subjects also reported at this time that they knew the tracing puzzles were mathematically impossible. Their results for the tracing puzzles were similarly discounted.

5.3 Results

A two-way analysis of variance (ANOVA) on the persistence on the insolvable puzzles data (Summarized in Table 5.1) showed no main effects for either the success-failure or the posture manipulations, $F(2,57) < 2, p < .2$ and $F(2,57) < 3, p < .07$ respectively. The analysis did reveal a statistically significant interaction effect, $F(2,57) = 4.1, p < .05$. These results are consistent with Riskind's findings. Further simple effects analysis by success-failure outcome revealed that success subjects exhibited more persistence when they used RoCo in its upright position ($M = 11.97$) after their success than when they used RoCo in its slupmed position ($M = 8.15$) or its neutral position ($M = 8.32$), $F(2,57) = 7, p < .01$. However, unlike Riskind's study, failure subjects showed no statistical difference across postures, $F(2,57) = 0.1$. The discussion section will hypothesize a few explanations for this difference.

Outcome	Slumped	Neutral	Upright
$n =$	*5*	*6*	*3*
Success	6.2	7.13	6.67
$n =$	*11*	*8*	*10*
Failure	5.64	5.63	5.8

Table 5.2: Average number correct on Remote Associates Test

Outcome	Slumped	Neutral	Upright
$n =$	*11*	*11*	*9*
Success	2.00	1.55	1.78
$n =$	*11*	*10*	*10*
Failure	1.55	2.00	1.6

Table 5.3: Average number of Tangrams solved in 7 minutes

For the Remote Associates Test, the analysis revealed no significant main effects or interaction effects. For the analysis, I removed the results of non-native English speakers and also the results of subjects who took the persistence task before this task as the persistence task generates many negative emotions like frustration and anger, thus likely undoing the success-failure manipulation. The data are summarized in Table 5.2. For the success-failure main effect, the analysis produced an $F(1, 37) = 1.43, p < .24$. For posture main effect, the analysis yielded $F(1, 37) = 0.12, p < .9$, and for the interaction effect, $F(1, 37) = 0.08, p < .93$.

For the Tangram test, ANOVA analysis on the number of tangram puzzles solved also showed no significant main effects or interaction effects. The data are shown in Table 5.3. For all three effects, $Fs < 1$. On the other hand, analyzing the time it took to solve the first tangram showed a significant success-failure main effect, $F(1, 50) = 6.1, p < 0.02$, but no other effects. The summarized data for these results are shown in Table 5.4

Outcome	Slumped	Neutral	Upright
$n =$	*11*	*10*	*8*
Success	107.27	176.9	165.75
$n =$	*8*	*8*	*11*
Failure	208.38	227.75	230.18

Table 5.4: Average time to solve first Tangram is seconds

5.4 Discussion

A number of factors have changed from the original study to adapt it to RoCo. These changes have a number of implications.

First, RoCo the robot is responsible for posture manipulation instead of a human experimenter. This change makes the manipulation more subtle and more unobtrusive. However, this subtlety comes at the cost of control. The user is free to adopt any posture he wishes as long as he can still read the screen. The expectation was that the screen readability constraint would be enough to manipulate the user to the desired posture. In practice, it appears that other factors like comfort also played a role.

Second, because RoCo can effectively be made "blind" to the user's posture and affective state, the dependent measure tasks can be performed while the subject is in the assigned posture. In Riskind's study, subjects were taken to a separate room and told to hold the assigned posture for approximately eight minutes under the pretense of a biofeedback experiment. They then performed the second set of tasks without controlling for posture. The expectation was that the mitigating or reinforcing effects of posture will be more pronounced in this study due to the near zero latency between posture manipulation and measured performance. Because the results failed to exhibit the "stoop to conquer" effect in all conditions, the validity of this expectation cannot be verified. However, in the one condition that showed significant results (success outcome with upright Roco posture), the effect does appear to be stronger than what Riskind found, $p < .01$ as opposed to $p < .03$ in Riskind's study.,

Using RoCo as a substitute for the experimenter in the study itself also reduces the danger of the experimenter inadvertantly introducing affective bias. Using the RoCo experiment system, RoCo can guide the user with instruction screens through each step of the experiment giving the user both privacy and avoiding experimenter contamination. A disadvantage of this approach is that RoCo cannot detect if the user does not understand the task and it cannot answer questions.

To limit the variability in the experiment, subjects were asked not to adjust their

chairs after the initial calibration. Several subjects asked or attempted to adjust the chair despite receiving instructions not to earlier. Additionally, many of the subjects attempted to adjust RoCo's monitor. This behavior suggests that posture manipulation was difficult to achieve, as users would only change their posture as a last resort when they could not adjust the chair or the monitor.

One reason users attempted to change RoCo's monitor position is that they did not recognize it as a robot, as another autonomous being. Many people thought that RoCo was simply a regular computer. As a result, they exhibited no mirrorring behaviors and felt more inclined to change its position before theirs. It would be interesting to see if introducing movement to RoCo would make posture manipulation easier by causing the user to adopt the mindest "I'm collaborating with a robot" instead of "I'm working on a machine."

Furthermore, although Riskind ruled out comfort level as a possible cause of his observations, comfort is certainly a factor in this case where RoCo can only suggest and hint at postures for the user to adopt. The posture that the user ultimately adopts will be determined in large part by comfort and less by affective state. A qualitative viewing of the subject footage also appears to bear out this comfort hypothesis. Another departure from Riskind's study is that subjects are free to change their posture at anytime, and indeed many subjects did change their posture, sometimes quite frequently.

The question remains, however, why did the success-upright condition show the effects of congruous posture so clearly while the other conditions were statistically indistinguishable? From the comfort hypothesis posulated above, it is possible that using RoCo in the upright state led to more comfortable upright posture from the user. The slumped state, in contrast, kept subjects in a slumped posture only as long as they needed to be on account of the discomfort. Thus, in the success-upright condition, comfort and natural tendancy aligned to produce the effect. In the slumped conditions, on the other hand, discomfort caused people to adopt other or more varied postures. Finally in the failure-upright case, it was the natural tendancy to slump after failure that caused people to hold their posture less.

Outcome	Slumped	Neutral	Upright
$n =$	*12*	*11*	*11*
Success	3.08	4.18	4.18
$n =$	*12*	*10*	*12*
Failure	3.08	4.10	3.25

Table 5.5: Average self-reported comfort levels

Indeed, ANOVA analysis of self-reported comfort levels shows a significant posture main-effect, $F(2, 62) = 4.12, p < 0.02$ (See Table 5.5). As one would expect, slumped postures are rated as less comfortable. Suprisingly, in the failure-upright case, comfort levels were as low as the slumped posture conditions. One explanation is that the low comfort levels were the result of a conflict between the tendancy to slump following failure with the manipulation of the monitor position that made an otherwise comfortable position uncomfortable.

For the Remote Associates Test, a quarter of the data could not be used because of the sequence effect. Subjects who received the persistence task before the Remote Associates Test, effectively experienced a negative affect manipulation, transforming success subjects into failure ones. In discounting those results, the data showed the beginnings of an outcome main effect, which is expected based on Isen's results, $F(1, 37) = 1.43, p < .24$. Unfortunately, after removing the invalid results, the data on the success side became very sparse and thus very sensitive to outliers, making further studies necessary to reinforce the results. As reported in the Results section, there appears to be no interaction effect, $Fs < 1, p < .9$.

The same comfort hypothesis might expalin this result as well. The video footage shows that users seem to adjust their posture for comfort, particularly while thinking about possible solutions. And while thinking, the primary posture manipulation technique (sceen viewability), is far less effective.

Additionally, Isen's study examined elation from receiving a gift as the affective state while this study attempts to produce feelings of success and failure. The difference in emotional states may also explain why the main effect is not showing as strongly as predicted.

Chapter 6

Conclusion

6.1 A New Research Platform

RoCo is an entirely novel social robot system that's designed to both promote back health and to manage the user's emotion to improve task and learning efficacy. Many of RoCo's subsystems are built on top of or as extensions of proven and reliable components from the Robotic Life Group and Affective Computer Group, and others at the MIT Media Lab. Thus, many of RoCo's subsystems are robust and well tested.

RoCo has also been designed with extensibility in mind. Because its input system is designed around IRCP, RoCo can support over 200 additional input sensors. Furthermore, RoCo's behavior system, the c5m architecture, is another highly extensible system for social behavior.

To support RoCo as a research platform for human subject experiments, I've also developed an experiment framework in Java. The framework has been tested on a 71 person study. This baseline user study that showed even in the static unmoving state, RoCo is capable of manipulating user posture to a certain extent. The results of this study will serve as a comparison for later studies which will further explore RoCo's potential.

6.2 Future Work

Many hours of posture video footage needs to be objectively and thoroughly reviewed. An initial pass through the footage by me suggests that valuable information on posture manipulation can be gleaned. It would be interesting to see what types of postures people adopted as well as how long they adopted them for.

Another important next step for RoCo is to develop a set of animations and behaviors which will allow more complex follow up studies.

On the hardware side, RoCo needs a stand or table to reside on. It would also benefit from connectors instead of screw terminals, which will make RoCo more robust and slightly more portable. Another nice feature would be brakes of some kind to prevent crashing as the result of sudden power loss or disabling of motors.

6.3 Follow Up Studies

There are a number of logical follow-ups to the baseline study presented in this thesis. One possibility is to explore the comfort hypothesis which suggests that comfort is the primary determinant for the postures users adopt during computer use. According to this hypothesis a less exaggerated RoCo slump state might allow the user to adopt more comfortable slumping postures and more effectively produce the "stoop to conquer" effect.

Another follow-up study would be to expose the user to RoCo's dynamic movement. By allowing the user to observe RoCo moving, he or she will be more likely to think of RoCo more as a robot than a computer. Once the user thinks of RoCo as an agent of some kind, he or she will also be more likely to mirror RoCo's posture. In conjuction with monitor viewability, the mirroring tendency might help overcome the discomfort factor.

Following up the creativity dependent measure is also an option. RoCo's movement could be used to induce various affective states in its user. One possibility is a social reference type of scenario where the user is presented with ambiguous or

relatively neutral results and RoCo's behavior is used to induce positive or negative affect, with the hope that the user will use RoCo as a social reference.

Appendix A

Trace Puzzles

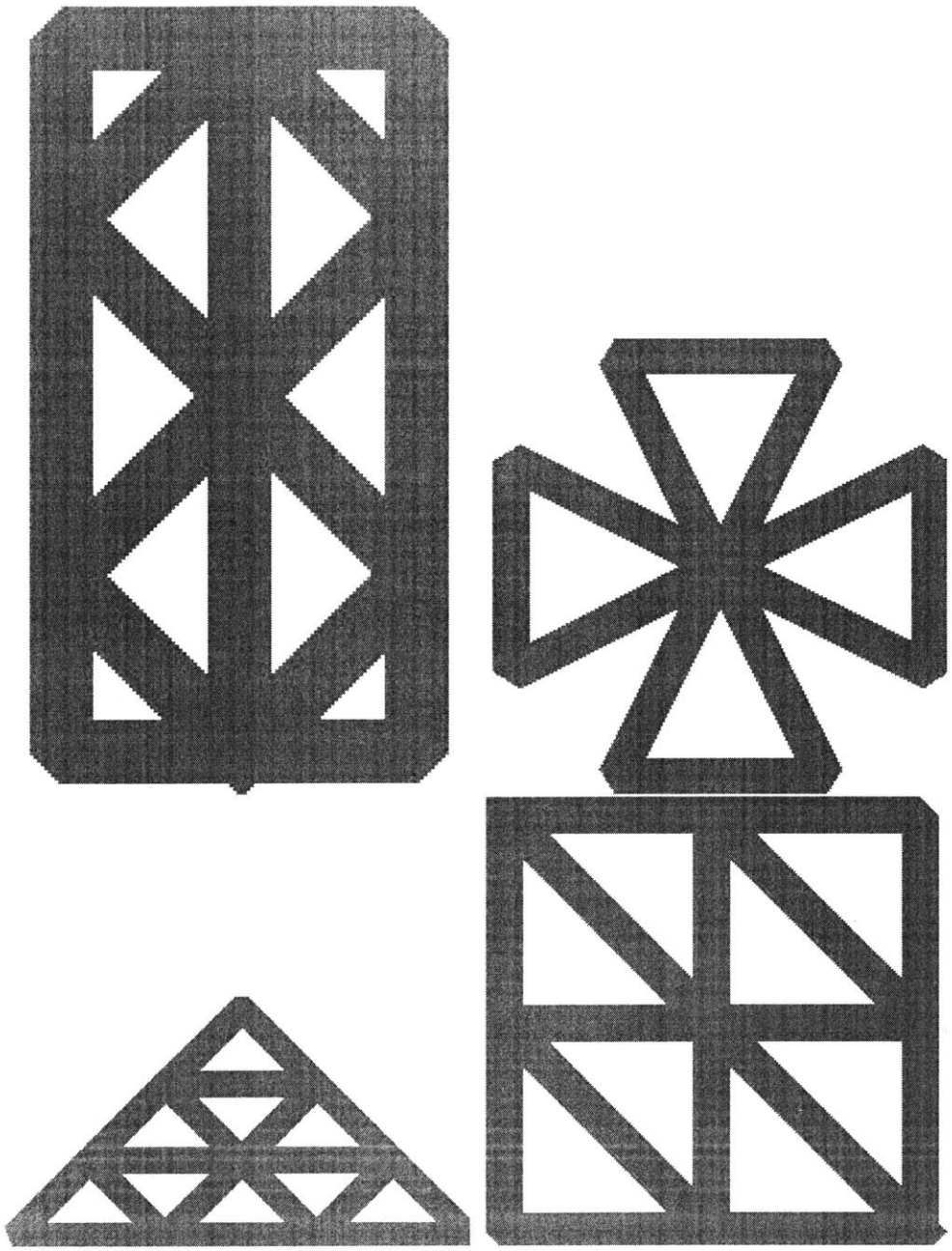

Figure A-1: Success Outcome: All solvable

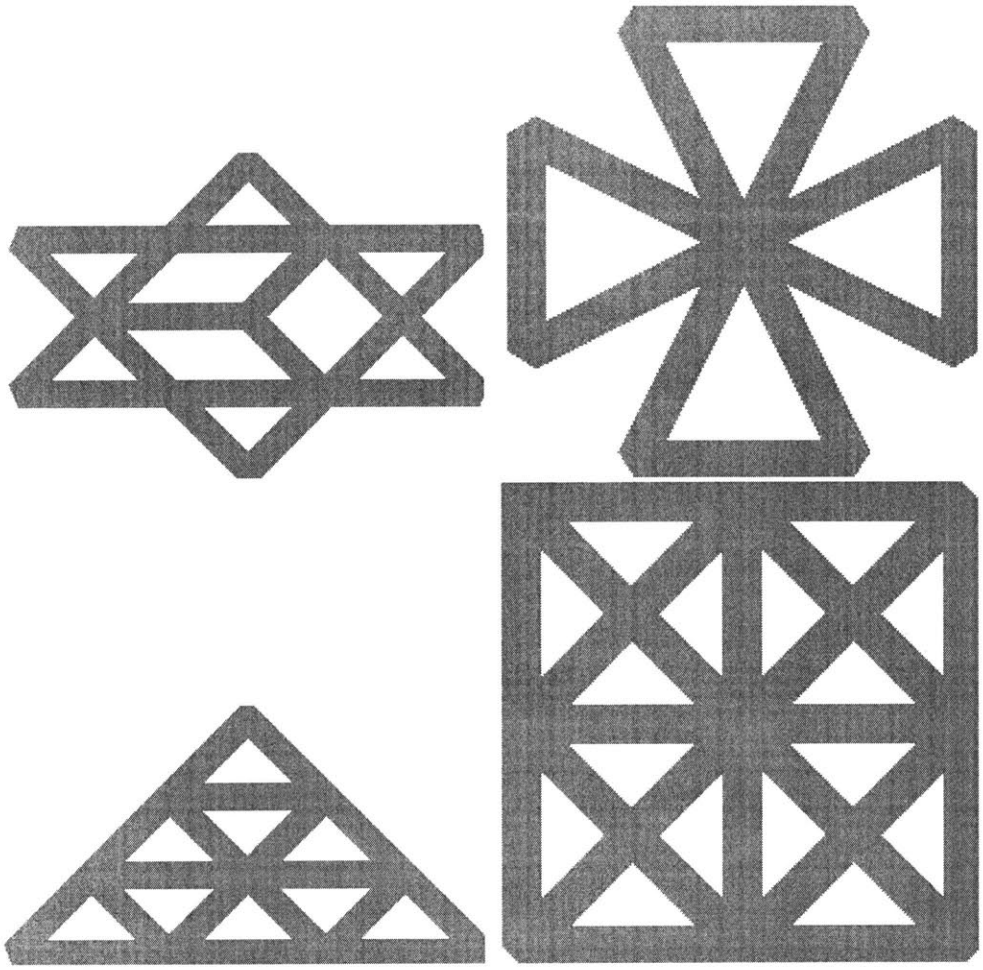

Figure A-2: Failure Outcome: Unsolvable, solvable, solvable, unsolvable

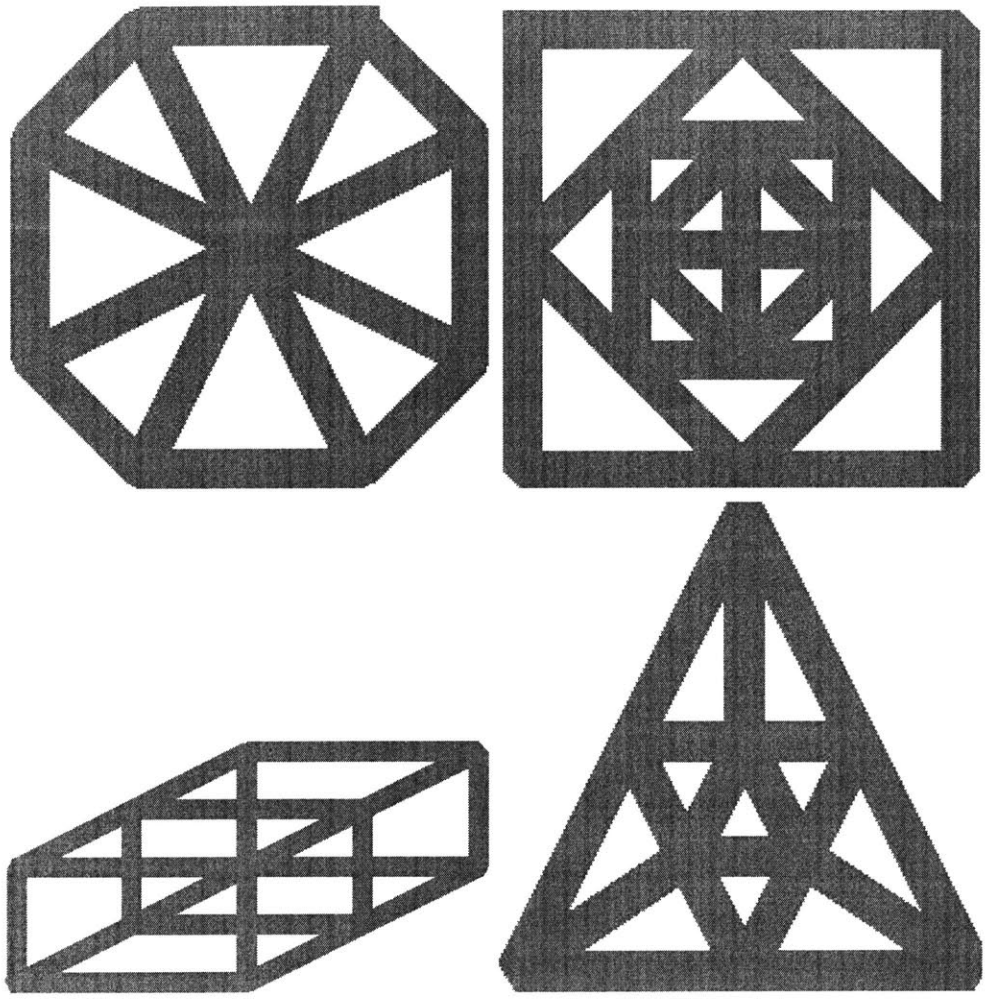

Figure A-3: Persistence Task: All unsolvable

Appendix B

Tangram Puzzles

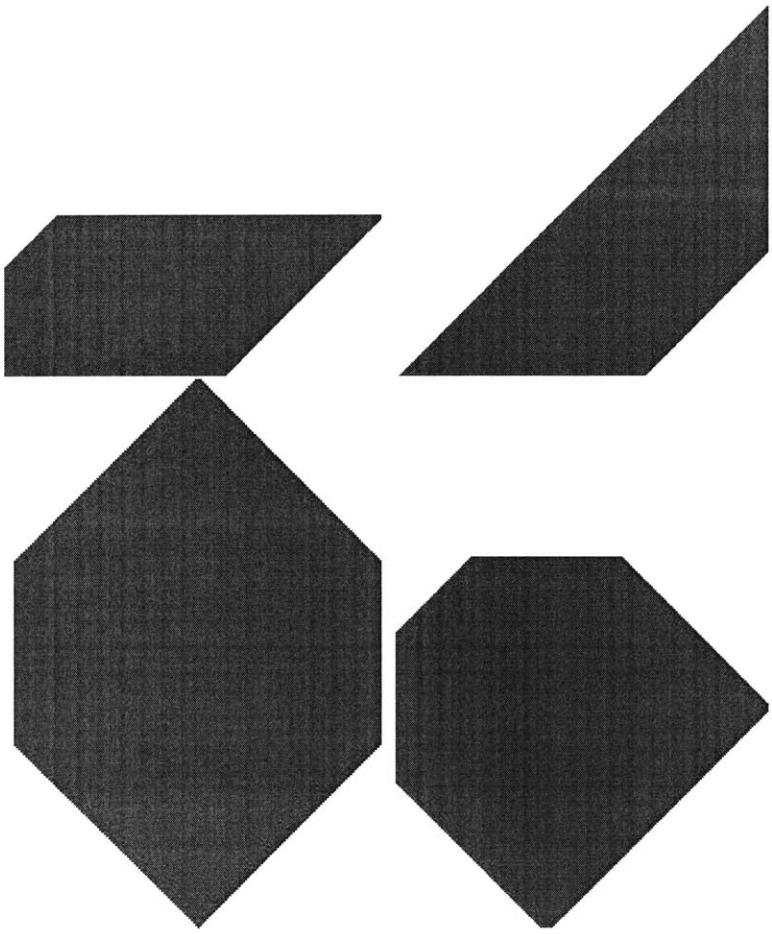

Figure B-1: Tangram puzzles: First four

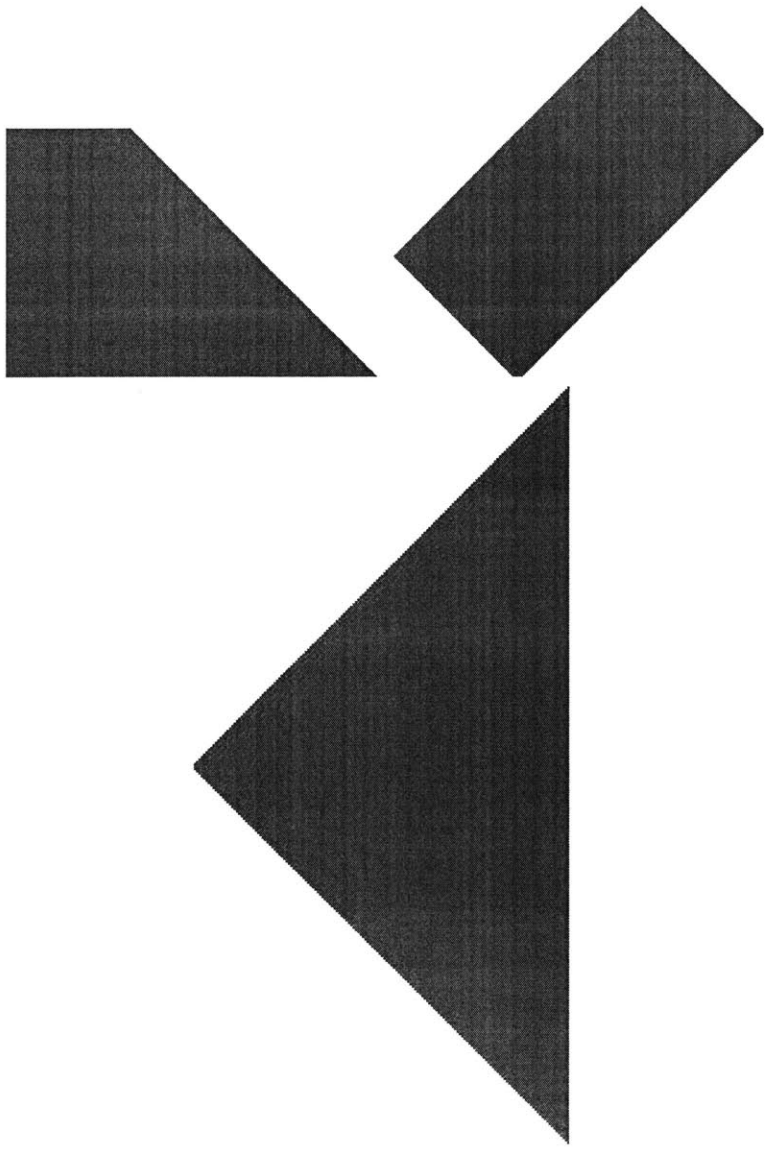

Figure B-2: Tangram puzzles: Last three

Appendix C

Remote Associates Test

Prompt	Answer
shelf,read,end	book
keel,show,row	boat
cookies,sixteen,heart	sweet
athletes,web,rabbit	foot
walker,main,sweeper	street
car,swimming,cue	pool
soap,shoe,tissue	box
desert,ice,spell	dry
inch,deal,peg	square
chamber,staff,box	music
base,show,dance	ball
jump,kill,bliss	joy
shopping,washer,picture	window
bass,complex,sleep	deep

Figure C-1: Remote Associates Test

Appendix D

Configuration

D.1 Motor configuration

Below is a listing of the configuration file `roco.dat` used to configure RoCo's motor control system.

```
name = "RocoMotorSystem"
description = "RoCo Motor System"

MEIXMP {
    type = "MEIXMP"
    description = "RoCo Controller"

    HeadYaw {
        name = "headRoll"
        description = "head yaw"
        motion_number = 0
        axis_number = 0
        filter_number = 0
        motor_number = 0
        motor_type = 0
        amp_polarity = 1
        pgain = 75
        igain = 2
        dgain = 55
        imax_moving = 0
        imax_idle = 10000
        drate = 7
        range = 43000
        invert_angles = 1
```

```
    min_angle = -0.523
    max_angle = 0.523
    pos_hw_action = 3
    pos_hw_polarity = 0
    pos_hw_direction = 0
    pos_hw_duration    = 0.1
    neg_hw_action = 3
    neg_hw_polarity = 0
    neg_hw_direction = 0
    neg_hw_duration = 0.1
    velocity = 3000
    acceleration = 5000
    deceleration = 5000
}

HeadPitch {
    name = "headNod"
    description = "head pitch"
    motion_number = 1
    axis_number = 1
    filter_number = 1
    motor_number = 1
    motor_type = 0
    amp_polarity = 1
    pgain = 70
    igain = 1
    dgain = 40
    imax_moving = 0
    imax_idle = 10000
    drate = 7
    range = 50800
    invert_angles = 1
    min_angle = -1.2
    max_angle = 0.25
    pos_hw_action = 3
    pos_hw_polarity = 0
    pos_hw_direction = 0
    pos_hw_duration    = 0.1
    neg_hw_action = 3
    neg_hw_polarity = 0
    neg_hw_direction = 0
    neg_hw_duration = 0.1
    velocity = 3000
    acceleration = 5000
    deceleration = 5000
```

```
}

HeadRoll {
    name = "neckPan"
    description = "head roll"
    motion_number = 2
    axis_number = 2
    filter_number = 2
    motor_number = 2
    motor_type = 0
    amp_polarity = 1
    pgain = 70
    igain = 1
    dgain = 50
    imax_moving = 0
    imax_idle = 10000
    drate = 7
    range = 74500
    min_angle = -1.25
    max_angle = 1.25
    pos_hw_action = 3
    pos_hw_polarity = 0
    pos_hw_direction = 0
    pos_hw_duration    = 0.1
    neg_hw_action = 3
    neg_hw_polarity = 0
    neg_hw_direction = 0
    neg_hw_duration = 0.1
    velocity = 4000
    acceleration = 5000
    deceleration = 5000
}

BasePitch {
    name = "baseTilt"
    description = "base pitch"
    motion_number = 3
    axis_number = 3
    filter_number = 3
    motor_number = 3
    motor_type = 0
    amp_polarity = 1
    pgain = 70
    igain = 1
    dgain = 30
```

```
    imax_moving = 0
    imax_idle = 10000
    drate = 7
    range = 51800
    invert_angles = 1
    min_angle = -0.2
    max_angle = 1.1
    pos_hw_action = 3
    pos_hw_polarity = 0
    pos_hw_direction = 0
    pos_hw_duration    = 0.1
    neg_hw_action = 3
    neg_hw_polarity = 0
    neg_hw_direction = 0
    neg_hw_duration = 0.1
    velocity = 4000
    acceleration = 5000
    deceleration = 5000
}

BaseYaw {
    name = "basePan"
    description = "base yaw"
    motion_number = 4
    axis_number = 4
    filter_number = 4
    motor_number = 4
    motor_type = 0
    amp_polarity = 1
    pgain = 110
    igain = 2
    dgain = 70
    imax_moving = 0
    imax_idle = 10000
    drate = 7
    range = 104118
    min_angle = -1.52
    max_angle = 1.52
    pos_hw_action = 3
    pos_hw_polarity = 0
    pos_hw_direction = 0
    pos_hw_duration    = 0.1
    neg_hw_action = 3
    neg_hw_polarity = 0
    neg_hw_direction = 0
```

```
        neg_hw_duration = 0.1
        velocity = 4000
        acceleration = 5000
        deceleration = 5000
    }
}
```

Variable	Description
name	Motor name
description	Plain text description
motion_number	MPI Motion supervisor number
axis_number	MPI Axis number
filter_number	MPI Filter number
motor_number	MPI Motor number
motor_type	MPI Motor type, 0 = servo, 1 = stepper
amp_polarity	Amplifier enable polarity, 1 = positive
pgain	Proportional gain value
igain	Integral gain value
dgain	Derivative gain value
imax_moving	Max amount of integral gain to use during motion
max_idle	Max amount of integral gain to use during idle
drate	Derivative gain sampling rate (0-7)
range	Encoder range in encoder counts
invert_angles	Flag that indicates whether model dof is the inverse of the physical do
min_angle	Minimum DOF angle from the model in radians
max_angle	Maximum DOF angle from the model in radians
pos_hw_action	Specifies the action to take when positive hardware limit is switched
pos_hw_polarity	Specifiecs the polarity of the positive hardware limit switch
pos_hw_direction	Whether to account for direction when triggering positive hardware switch
pos_hw_duration	Minimum amount of time in seconds before switch triggers
neg_hw_action	Specifies the action to take when negative hardware limit is switched
neg_hw_polarity	Specifiecs the polarity of the negative hardware limit switch
neg_hw_direction	Whether to account for direction when triggering negitive hardware switch
neg_hw_duration	Minimum amount of time in seconds before switch triggers
velocity	Velocity in encoder counts per second
acceleration	Acceleration in encoder counts per second squared
deceleration	Deceleration in encoder counts per second squared

Table D.1: Motor Configuration variables

D.2 Experiment Launcher Configuration

Below is a listing of an sample experiment configuration xml file. It has one experiment configuration called "condition 1" that executes a `DemographicExperiment`, a `QuicktimeExperiment`, and a `TracePuzzleExperiment`.

```
<user-study>
    <config name="condition1">
        <controller class="drew.roco.DemographicExperimentController">
         <param type="string" value="contentroot/robots/roco/user_study/part1_instr.html"/>
        </controller>

        <controller class="drew.roco.QTExperimentController">
                <param type="string" value="contentroot/robots/roco/user_study/neutral.avi"/>
                <param type="string" value="contentroot/robots/roco/user_study/neutral_instr.html"/>
        </controller>

        <controller class="drew.roco.TracePuzzleController">
            <param type="string array" value="contentroot/robots/roco/user_study/soluble1.txt,
                                              contentroot/robots/roco/user_study/soluble2.txt,
                                              contentroot/robots/roco/user_study/soluble3.txt,
                                              contentroot/robots/roco/user_study/soluble4.txt"/>
            <param type="string" value="contentroot/robots/roco/user_study/trace_instr.html"/>
            <param type="string" value="contentroot/robots/roco/user_study/success.html"/>
            <param type="int" value="120"/>
        </controller>
    </config>
</user-study>
```

Appendix E

RoCo Wiring Setup

This section details RoCo's wiring setup. It's meant to serve as a quick reference for the way RoCo is currently configured. You should familiarize yourself with the MEI control card hardware manual before undertaking any significant changes. The manual is available online through MEI's website.

E.1 Encoders

The encoder signal for each motor must be split between the servoamplifier and the MEI control card. In addition, the MEI control card is designed for differential encoders not signle-ended. To adjust for this, there must also be a bias circuit. See Figure E-3 for wiring details. RoCo uses the TTL bias circuit configuration. Also, the encoders draw power from the servoamplifiers and not the MEI control card.

E.2 Limit Switches

RoCo's limits switches can be configured either in "off" or "on" state. That is, "off" can be considered triggered or not. "Off" is currently mapped to untriggered. `HomeLim_Rtn` is tied to a `5V_Out` terminal. The black wire is connected to `Gnd` and the brown wire is connnected to `Neg_Lim_IN` or `Pos_Lim_IN` depending on which limit switch is being wired. See Figure E-2 for the corresponding terminal numbers.

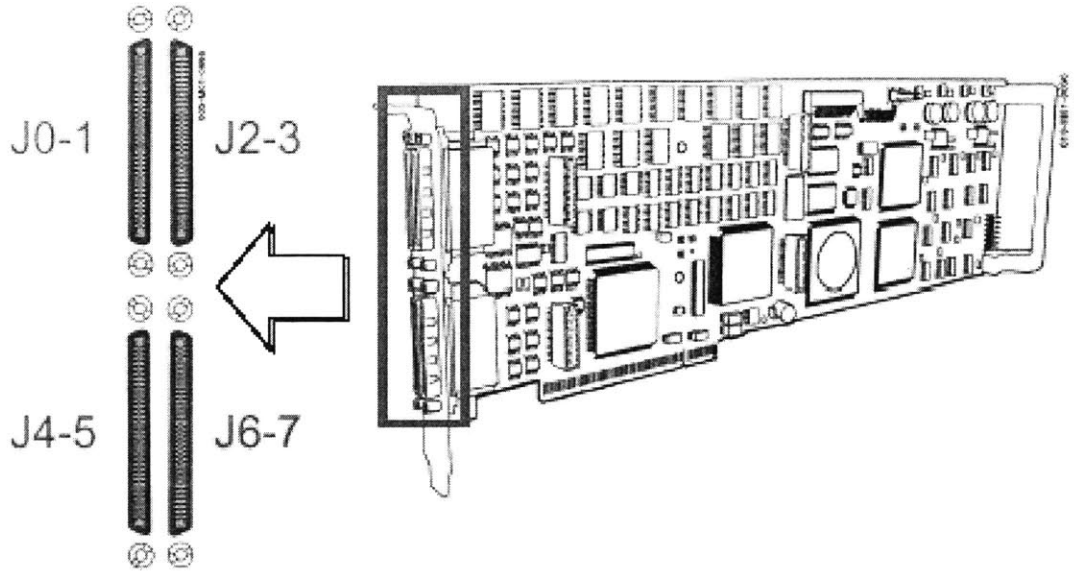

Figure E-1: A diagram of the PCI control card

E.3 Motor card Output

The output from the motor control card goes to the servoamplifiers. The `Cmd_Dac_OUT_+` and `Cmd_Dac_OUT_-` are connected to `+Set value` and `-Set value` servoamplifier terminals respectively.

Additionally, to allow the software to programmatically enable and disable the amplifiers, `Amp_En_Emitter` is connected to the Enable input of the amplifier while the `Amp_En_Collector` is connected to `5V_OUT`.

Again, see Figure E-2 for the corresponding terminal numbers.

E.4 Servoamplifier Setup

The Maxon servoamplifiers also require a bit of setup. In order for the servoapmlifiers to correctly detect and handle the encoder input, the number of pole-pairs must be set. The only way to set this value is through the RS232 interface and a Maxon provided utility application. This value only needs to be set once. It will be saved across power cycles. Finally, the servoamplifiers should be set in current control mode.

Pin	Signal	Signal	Pin
1	Analog_IN_0+	Analog_IN_0-	35
2	Analog_IN_1+	Analog_IN_1-	36
3	Gnd	AGnd	37
4	Enc0_A+	Enc0_A-	38
5	Enc0_B+	Enc0_B-	39
6	Enc0_I+	Enc0_I-	40
7	Home0_IN	5V_OUT	41
8	Pos_Lim0_IN	Gnd	42
9	Neg_Lim0_IN	HomeLim0_Rtn	43
10	Cmd_Dac_OUT_0+	Cmd_Dac_OUT_0-	44
11	Aux_Dac_OUT_0+	Aux_Dac_OUT_0-	45
12	Amp_Flt0_IN	Amp_Flt0_Rtn	46
13	Amp_En0_Collector	Amp_En0_Emitter	47
14	UserIO_A0	UserIO_A0_Rtn	48
15	Xcvr0A+	Xcvr0A-	49
16	Xcvr0B+	Xcvr0B-	50
17	Xcvr0C+	Xcvr0C-	51

Figure E-2: Pinout diagram for axis 0

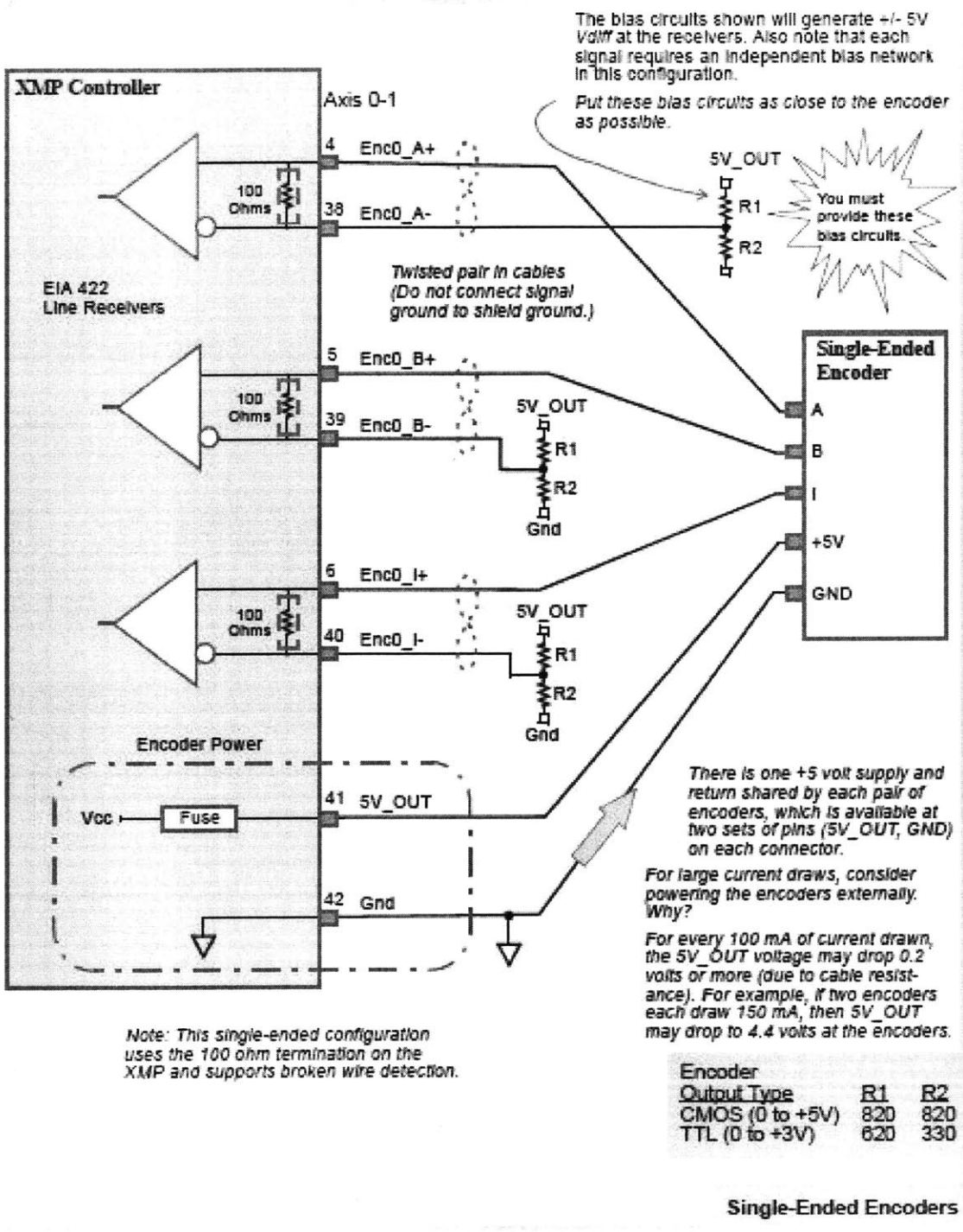

Figure E-3: Single-ended encoder wiring

Appendix F

Experiment Data

Below is the raw data that I collected from 71 subjects. The comments column will list a reason for empty cells. The page column denotes the order that the subjects took the tests in the second part. The ordering is below shown in Tabel F.1

Letter	Test Order
A	RAT, Tangram, Trace
B	RAT, Trace, Tangram
C	Tangram, RAT, Trace
D	Tangram, Trace, RAT
E	Trace, Tangram, RAT
F	Trace, RAT, Tangram

Table F.1: Page letter and corresponding test ordering

c1p1

Success-Neutral												
Page	Age	Gender	Comp Use	Education	Puzzle	RAT	Tangrams (n)	Tangrams (t)	Trace	Puzzle	Comfort	Comment
1A	23	M	6	2	5	5.00	2	164	9.25	4	4	
2B	22	F	4	1	4	9.00	1	197	11.25	3	4	
3C	23	M	6	1	4	6.00	3	108	8.5	2	6	
4C	20	M	6	1	5	8.00	2	190		6	5	Knew unsolvable
5E												Difficulty understanding
6F	18	M	7	2	4	7.00	1	83	4.5	3	4	
7A	22	F	5	2	3	8.00	1	58	6	2	5	
8A	23	F	5	2	4		3	102	14.5	3	4	Non native
9C	25	F	5	1	2	7.00	2	103	6	2	2	
10D	35	M	7	2	6		1	369	3.5	5	5	Non native
11E	31	F	7	1	6	2.00	0		11.75	5	6	
12F	24	F	7	2	4	3.00	1	395	10	4	1	Non native

Page 1

Figure F-1: Success-Neutral Data

c1p2

Success-Slumped												
Page	Age	Gender	Comp Use	Education	Puzzle	RAT	Tangrams (n)	Tangrams (t)	Trace	Puzzle	Comfort	Comment
1A	18	F	5	1	5	4	3	139	8.75	3	3	
2B	24	F	5	2	1	4	1	132	5.5	1	3	
3C	18	M	6	0	5	6	1	88	4.25	5	2	
4D	23	M	7	2	5	8			7	4	3	Tried to solve tangrams with pen
5E	27	M	7	1	5	8	4	57		5	3	Knew about unsolvability after first
6F	27	M	6	0	4	9	1	109	5	4	2	
7A	24	M	7	1	5		1	245	9	3	4	Non native
8B	21	F	5	1	6	9	0	102	10	6	4	
9C	22	M	5	1	5	8	4	78	2.5	4	4	"gave up"
10D	21	F	6	1	5		2	99	10.5	4	4	Non-native
11E	18	M	7	0	4	7	1	69	3.75	1	3	
12F	23	M	4	1	6		4	62	7.75	5	2	Non native

Page 1

Figure F-2: Success-Slumped Data

c1p3

Success-Upright

Page	Age	Gender	Comp Use	Education	Puzzle	RAT	Tangrams (n)	Tangrams (t)	Trace	Puzzle	Comfort	Comment
1A	22	F	7	2	3	1	1	139	6.5	2	4	
2B	24	F	6	2	6	11	2	79	12.75	4	4	
3C	21	F	6	1	5	8	4	111		5	3	Knew trace unsolvable
4D	21	F	6	1	4	1			17	5	2	Couldn't do tangrams
5E	22	F	6	1	4	7	2	140	12	4	3	
6F	22	M	5	1	4	7			2	2	6	Success-failure manip questionable
7A	25	M	7	2	5					3	5	Knew unsolvable, non-native
8B	24	F	7	1	2	2	2	185	13.25	2	6	
9C	22	M	7	1	5	2	1	85	9.5			
10D	22	M	7	1	4	7	1	358	11.25	3	5	
11E	18	F	6	2	2	2	3	229	7.25	1	5	
12F	22	F	6	2	4	7	0		9.25	3	3	

Figure F-3: Success-Upright Data

c2p1

Failure-Neutral												
Page	Age	Gender	Comp Use	Education	Puzzle	RAT	Tangrams (n)	Tangrams (t)	Trace	Puzzle	Comfort	Comment
1A	19	F	4	1	2	2	6	58	8.5	1	7	
2B	23	M	7	2	7	12	4	229	7.25	6	5	
3C												Didn't believe
4D												Didn't understand instructions
5E	28	M	7	2	5	8	1	360	7	4	4	
6F	22	F	5	2	5	4	4	115	7.75	4	4	
7A	22	F	3	1	4	2	1	402	8.25	3	3	
8B	22	F	6	1	3	3	2	148	7.25	2	5	
9C	19	F	6	1	5	8	0		5.5	4	4	
10D	32	F	7	2	4		1	136	7.75	3	4	Non-native
11E	25	F	5	2	5	6	1	374	11.75	4	4	
12F	20	F	7	1	2		0		6.5	1	1	Non-native

Page 1

Figure F-4: Failure-Neutral Data

c2p2

Failure-Slumped

Page	Age	Gender	Comp Use	Education	Puzzle	RAT	Tangrams (n)	Tangrams (t)	Trace	Puzzle	Comfort	Comment
1A	21	M	6	1	5	8	3	61	7.75	2	2	
2B	19	M	6	0	3	5	3	188	13.25	2	3	
3C	20	F	6	2	4	4			10.5	4	3	Tangram results invalid
4D	34	F	7	1	5	7	2	246	6.5	3	2	
5E	31	M	6	2	4		0		6.25	2	3	Non-native
6F	22	M	7	2	4	5	2	138	5	3	5	
7A	56	M	7	2	3	8	1	251	7.25	1	5	
8B	20	M	6	1	5	7	2	264	4	3	3	
9C	36	F	5	1	4	2	0		9.5	2	2	
10D	23	M	6	2	5	5	3	165	5.75	2	3	
11E	21	F	7	1	4	4	1	354	5.75	2	3	
12F	40	F	3	1	3	8.00	0		6.5	1	3	

Page 1

Figure F-5: Failure-Slumped Data

c2p3

Failure-Upright												
Page	Age	Gender	Comp Use	Education	Puzzle	RAT	Tangrams (n)	Tangrams (t)	Trace	Puzzle	Comfort	Comment
1A	22	M	6	1	5	9	2	55	6.5	4	3	
2B	26	M	6	2	5	9	2	69	6.25	2	3	
3C	21	F	4	1	3		1	405	11.25	2	2	RAT test invalid
4D	21	F	6	1	2	3	0	416	8	1	2	
5E	26	M	7	2	5	5		250	6	5	3	Trouble with tangrams
6F	18	M	7	0	6	6	1	58	8.5	5	3	
7A	20	F	6	0	4	7	3	102	6	2	4	
8B	36	M	7	1	5	5	4	348	5.75	4	5	
9C	34	M	2	2	2	2	1	381	8.25	2	6	
10D	24	M	5	2	4	6	1	393	5.75	3	3	
11E	26	F	5	2	5	6	1		9.25	2	2	

Page 1

Figure F-6: Failure-Upright Data

c1p1

Success-Neutral												
Page	Age	Gender	Comp Use	Education	Puzzle	RAT	Tangrams (n)	Tangrams (t)	Trace	Puzzle	Comfort	Comment
1A	23	M	6	2	5	5.00	2	164	9.25	4	4	
2B	22	F	4	1	4	9.00	1	197	11.25	3	4	
3C	23	M	6	1	4	6.00	3	108	8.5	2	6	
4C	20	M	6	1	5	8.00	2	190		6	5	Knew unsolvable
5E												Difficulty understanding
6F	18	M	7	2	4	7.00	1	83	4.5	3	4	
7A	22	F	5	2	3	8.00	1	58	6	2	5	
8A	23	F	5	2	4		3	102	14.5	3	4	Non native
9C	25	F	5	1	2	7.00	2	103	6	2	2	
10D	35	M	7	2	6		1	369	3.5	5	5	Non native
11E	31	F	7	1	6	2.00	0		11.75	5	6	
12F	24	F	7	2	4	3.00	1	395	10	4	1	Non native

Page 1

Figure F-7: Success-Neutral Data

Bibliography

[1] M. Argyle. *Bodily Communication.* Methuen and Co. Ltd., New York, NY, 1988.

[2] C. Breazeal. Designing sociable robots. 2002.

[3] Isla D. Downie M. Ivanov Y. Burke, R. and B. Blumberg. Creaturesmarts: The art and architecture of a virtual brain. In *Game Developers Conference*, pages 147–166, San Jose, CA, 2001.

[4] L. Chirstensen and K. Menzel. The linear relationship between student reports of teacher immediacy behaviors and perception of state motivation, and of cognitive, affective, and behavioral learning. *Communcation Education*, 47:82–90, 1998.

[5] Laird J. Schneider E. Sexter M. Stern L. Duclos, S. and O. Van Lighten. Emotion-specific effects of facial expressions and postures on emotional experience. *Journal of Personality and Social Psychology*, 57:100–108, 1989.

[6] F. Faiks and S. Reinecke. Investigation of spinal curvature while changing one's posture during sitting. *Contemporary Ergonomics*, 1998.

[7] J. Glass and J. Singer. *Urban Stress.* Academic Press, New York, NY, 1972.

[8] M Hancher. A motor control framework for many-axis interactive robots. Master's thesis, Massachusetts Institute of Technology, 2003.

[9] . Essa I. Haro, A and M. Flickner. Detecting and tracking eyes by using their physiological properties, dynamics and appearence. In *Proceedings of IEEE Computer Vision and Pattern Recognition*, Hilton Head, SC, 2000.

[10] W. Hartup. *Cooperation, Close Relationships, and Cognitive Development*. Cambridge University, Cambridge, 1996.

[11] Daubman K. Isen, A. and G. Nowicki. Positive affect facilitates creative problem solving. *Journal of Personality and Social Psychology*, 52:1122–1131, 1987.

[12] A. Kapoor and R. Picard. Real-time, fully automatic upper facial feature tracking. In *Proceedings of the 5th International conference on automatic face and gesture recognition*, 2002.

[13] Bevins T. Weisman J. Krag M.H. Reinecke, S. and M.H. Pope. The relationship between seating postures and low back pain. In *Rehabilitation Engineering Society in North Ameriac, 8th Annual Conference*, 1995.

[14] V. Richmond and J. McCroskey. *Immediacy, Nonverbal Behavior in Interpersonal Relations*. Allyn and Bacon, Boston, MA, 1995.

[15] J.H. Riskind. They stoop to conquer: Guiding and self-regulatory functions of physical posture after success and failure. *Journal of Personality and Social Psychology*, 47:479–493, 1984.

[16] J.H. Riskind and C.C. Gotay. Physical posture: Could it have regulatory or feedback effects upon motivation and emotion? *Motivation and Emotion*, 6:273–296, 1982.

[17] Ray R.R. Rottenberg, J. and J.J. Gross. *Emotion elicitation using films*. Oxford University Press, New York, NY, 2004.

[18] V.E. Wilson and E. Peper. The effects of upright and slumped postures on recall of postive and negative thoughts. *Applied Psychophysiology and Biofeedback*, 29:189–195, 2004.