A Physical Model for Tracking Human Seating Posture

by

Ifung Lu

B.S., Mechanical Engineering
Massachusetts Institute of Technology, 1997

Submitted to the Department of Mechanical Engineering
in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Mechanical Engineering

at the
Massachusetts Institute of Technology
June 1998

Author: ______________________
Department of Mechanical Engineering
May 8, 1998

Certified by: ______________________
Hong Z. Tan
Research Scientist, Media Laboratory
Thesis Supervisor

Certified by: ______________________
H. Harry Asada
Professor of Mechanical Engineering
Director, d'Arbeloff Laboratory for Information Systems and Technology
Thesis Reader

Accepted by: ______________________
Ain A. Sonin
Professor of Mechanical Engineering
Chairman, Graduate Thesis Committee

A Physical Model for Tracking Human Seating Posture

by

Ifung Lu

Submitted to the Department of Mechanical Engineering
on May 26, 1998 in Partial Fulfillment of the
Requirements for the Degree of Master of Science in
Mechanical Engineering

ABSTRACT

I present a three-dimensional model of a human in a seated posture that consists of two deformable thigh models connected to a rigid torso. I develop a framework for taking contact pressure data gathered from a real person sitting in a chair and, using the physics of the human model, compute the angles of inclination when the person leans in the chair. Preliminary results of the model's ability to reconstruct leaning posture are shown.

Thesis Supervisor:	Hong Z. Tan
Title:	Research Scientist, MIT Media Laboratory
Thesis Reader:	H. Harry Asada
Title:	Professor of Mechanical Engineering

ACKNOWLEDGMENTS

What to say… there are so many people that helped me through the course of working on this thesis.

I'd like to thank Hong Tan for being my thesis advisor.

I also want to thank Sandy Pentland and the Media Laboratory for financially supporting my research.

Thanks to my thesis reader Harry Asada for reading my thesis and also for being my liaison to the mechanical engineering department.

Thanks to Leslie Regan for her help with all those little details involved with submitting a thesis to the mechanical engineering department.

Thanks to Sumit Basu, colleague and roommie for a year, for his inspiration, his hate for ugly jackets and BPP. I will never forget that, damn it!

Thanks to Chris Wren for letting me use his BodyModel code in my thesis and for his help with understanding the system.

Thanks to Tony Jebara for teaching me about EM and allowing me to use his code in my thesis.

Thanks to Thad Starner for his help deciphering the Thingworld code even though it was, in my opinion, permanently DOA. Lisp and non-ANSI C suck.

Thanks to Tanzeem Choudhury for help with Matlab and all those other "little" details that I was getting caught up on.

Thanks also to all my friends who put up with me while I was "thesisizing": C.B., M.B., S.B., J.C., P.C., C.E., D.D., J.D., J.F., J.K., J.L., D.G., H.D., H.G., K.L., S.L., S.L., F.N., P.P., E.S., I.S., B.T., B.T., F.T., S.V., A.W., M.Y.

Last, but not least, thanks to my parents for everything. I love you both very much.

TABLE OF CONTENTS

ABSTRACT....2

Chapter 1: Introduction 6

1.1 MOTIVATION FOR THE WORK....7

1.2 PREVIOUS WORK....9

Chapter 2: Methodology and System Architecture 12

2.1 OVERVIEW OF THE SEATING POSTURE TRACKING SYSTEM....12

2.2 PREPROCESSING USING EXPECTATION MAXIMIZATON....14

2.3 THE FINITE ELEMENT METHOD AND THINGWORLD....19

2.4 A MODULAR CONSTRAINT SYSTEM AND BODYMODEL....23

Chapter 3: Testing and Results 25

3.1 TRAINING THE SYSTEM....25

3.2 PERFORMANCE OF THE SYSTEM....26

Chapter 4: Discussion 27

APPENDIX A: DATA PREPROCESSING CODE....30

APPENDIX B: MODIFIED THINGWORLD CODE....33

APPENDIX C: BODYMODEL CODE....37

APPENDIX D: PARALLEL VIRTUAL MACHINE....39

TABLE OF FIGURES

Figure 1 – SPoTS component architecture 12
Figure 2 – SPoTS running on a Silicon Graphics machine 13
Figure 3 – Tekscan 42x48 pressure map 14
Figure 4 – EM fitting of pressure maps 17
Figure 5 – Resampling of the pressure data 18
Figure 6 – Geometry of the thigh models 19
Figure 7 – 27-node isomparametric finite element 20
Figure 8 – Some modes of vibration of the thigh model 22
Figure 9 – Force balance 23
Figure 10 – The puppet leaning to the right 24

Chapter 1

INTRODUCTION

One of the greatest barriers to automated computer recognition and understanding of human behavior is the lack of sensory "organs" in today's personal computers. Even the most powerful computer running the most advanced software cannot see, hear, taste, smell, or touch objects in their environment without sensors. Yet we implicitly expect them to be aware of these things and react intelligently. While this is changing for computer vision and speech recognition as video cameras and voice microphones are finding their way to consumers and are slowly becoming mainstream peripherals, haptic, or touch-based input and output devices are only now just starting to appear on the information landscape.

I propose and implement a system that tracks the leaning state of a seated person from the contact pressure distribution, i.e., it will "feel" the pressure of the person sitting on the sensor and report that he or she is leaning to the left 10 degrees and forward 15 degrees. The Seated Posture Tracking System (SPoTS) takes the contact pressure data obtained using the Tekscan pressure sensor system and employs a physical model of a seated human to calculate two inclination angles that characterize how the person is leaning in the chair. This model "interprets" the contact pressure data in a physically interpretive way by approximating the physics involved in the body while leaning in the chair.

1.1 Motivation for the Work

An important question to answer about this work is why we want to track human seated posture? One answer is that deriving the seating posture from the contact pressure between the human and the chair in a physically interpretive way is an interesting research problem that requires a lot of mechanical engineering. This problem can be solved in a variety of non-mechanical ways, such as training a purely statisitcal system like a neural network or a hidden Markov model. However, we believe that taking a physical approach can provide certain advantages in terms of the ease of understanding and extending the system. Because we use a physical model, every step in the process that we outline in the next chapter can be justified and interpreted in a physical way. In other words, instead of maximizing the likelihood or probability of an event as we would in a statistical framework, we work with balancing real forces and minimizing strains in the physical realm. In addition, because we do work in the physical domain, extending the system involves simply connecting new structures and control schemes or applying new forces to our model.

Another reason for solving this problem is that we are providing a methodology and a toolkit for tracking human seating posture. By providing a method to calculate leaning posture from the contact pressure, we can then focus our attention on using this in a useful way. Two possible applications that could potentially take advantage of posture tracking are in the areas of driver safety and chair ergonomics.

In the area of driver safety, posture tracking can assist a system that is trying to learn the relationship between posture changes and driver intention by allowing it to look at inclination angles instead of raw pressure data. It has long been recognized that driver intention is linked to real, physical manifestations of these intentions. For example, one of the first things a driver education teacher warns the driving student is to avoid veering the car to the left while checking for obstacles in the left rearview mirror. If a predictive relationship between the driver's posture and his or her intention to speed up, slow down, or change lanes could be established, then the car would be aware of the driver's intentions and could do something to warn people of potential collisions and other hazards. Although having a posture tracking system would not directly solve this learning problem, it could assist a learning system by providing it a set of physically-based guesses of what it thinks the leaning state is, based on the pressure data.

Another instance in which having a method for posture tracking would be beneficial is in the area of chair ergonomics. People need to move in the chair from time to time to redistribute the stresses on body parts that come into contact with the chair. Usually, we do this naturally without thinking about it very much. However, sometimes when we are really concentrating on performing a task, we override our natural control systems and neglect to shift positions. This can lead to discomfort in the chair or even the dreaded "pin-cushion" sensation. By tracking the seated posture, we can remind the person to move from time to time if he or she has not been doing so, with an ergonomically sensitive "active-comfort" system, if you will. Again, the posture tracker would not solve the ergonomics problem directly. However, it does provide physically interpretive data in the form of leaning angles to the "active-comfort" system so that it can react appropriately.

We believe that using pressure sensors and posture tracking software is important because it opens up access to a different kind of information not accessible to other sensors. Pressure sensors can complement, and in some circumstances, replace the use of other sensors like cameras. An example of this involves the "Smart Desk" research project being conducted at the Media Laboratory at the Massachusetts Institute of Technology. This research involves tracking the upper-body in three dimensions with a pair of cameras. Recently, some people have been interested in extending the system with the capability to track the lower body as well. Because the table completely occludes the line of sight to the lower body, a traditional vision solution to the tracking problem would involve placing one or more cameras under the table. However, when doing so, one quickly encounters problems with lighting, chair occlusion and body self-occlusion that are associated with vision-based solutions. By using a pressure sensor and a posture tracking system, we avoid many of these problems altogether. We do not have to worry about lighting the scene because we are sensing contact pressure and not light. Although, in some sense, it can be said that the bottom the body that is in contact with the chair completely occludes the "view" of the rest of the upper body, the model can calculates the most likely upper body leaning position that is be responsible for the detected contact pressure distribution.

1.2 Previous Work

In looking at the prior work, we've come to realize that not much has been done in terms of using a physical model to track human seating posture. Most of the related work generally involves the modeling of other body parts like the lips and the fingertips. Also, the approach taken by previous work is divided mostly into two camps, with statistically based systems that focus on computation speed on one side, and slow, highly accurate biomechanical models on the other. We hope to bridge this gap by providing a reasonably accurate system that can compute leaning angles from contact pressure in real time.

Tan, Lu and Pentland [18] have constructed an "eigenposture" system that employs a statistical method to do human posture classification of contact pressure data. The eigenposture system uses a method similar to the Turk and Pentland face recognition system [19]. The eigenposture system is trained initially on pressure data for each of the postures to be recognized. The data set is reduced to a set of eigenvectors that characterize the "modes of vibration" of the posture space and together, span the space that covers all the training data, i.e., these eigenvectors can be linearly combined to form any pressure map on which it was trained. Each of the trained postures is associated with a unique point of the posture space. In the recognition phase, the system projects the new data onto this eigenspace. The system then finds the distance from the projection to the points corresponding to the various trained postures. It classifies the new data with a posture label if the distance from its projection is the minimum, and is within a predetermined threshold. This system sacrifices some accuracy for speed by using only eigenvectors associated with the largest eigenvalues.

Pawluk [13] and Dandekar [5], on the other hand, instead of training a model using statistical methods, take a mechanical approach and explicitly model the fingerpad structures with biomechanical data. Pawluk has created a quasi-linear viscoelastic model of the human fingerpad that is subjected to Hertzian contact. Dandekar [5] has assembled a three-dimensional finite element model of the material layers in the fingertip using the large displacement formulation. The FEM model is solved on a supercomputer for the strain and stress state. Both approaches sacrifice speed for high accuracy of the results.

Basu [1] takes an interesting approach in tracking human lip movement with a hybrid system that employs a simple finite element lip model as a "physical prior" for a probabilistic tracking system based on skin-color. The physical model imposes physical constraints that regularize between sparse observation points to derive the correct physical modes for the model.

We hope to emulate the approach taken by Basu to find a happy compromise between high speed and high numerical accuracy. We want to construct a system that is "accurate enough" for posture tracking and computationally simple enough to run in real-time. We approach this problem by making a quasi-static assumption about the dynamics of the system, and by approximating each thigh as a single, homogeneous elastic object, each fitted by a single, three-dimensional isoparametric finite element. We also make some assumptions about the positioning of various body parts as well as the range of motion the system can track. Making these simplifying assumptions allow us to track human seated posture by calculating leaning angles from contact pressure data in realtime.

Chapter 2

METHODOLOGY AND SYSTEM ARCHITECTURE

2.1 Overview of the Seating Posture Tracking System (SPoTS)

The Seating Posture Tracking System (SPoTS) is a family of components programmed in ANSI C and C++ and in the Matlab 5.0 scripting language for use on the Silicon Graphics family of UNIX workstations.

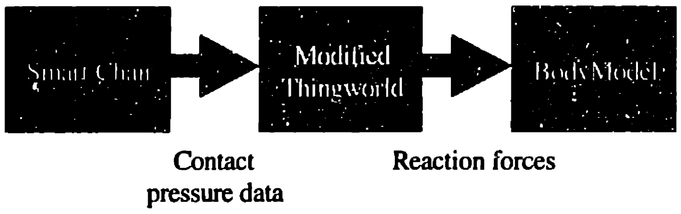


Figure 1 - SPoTS component architecture.

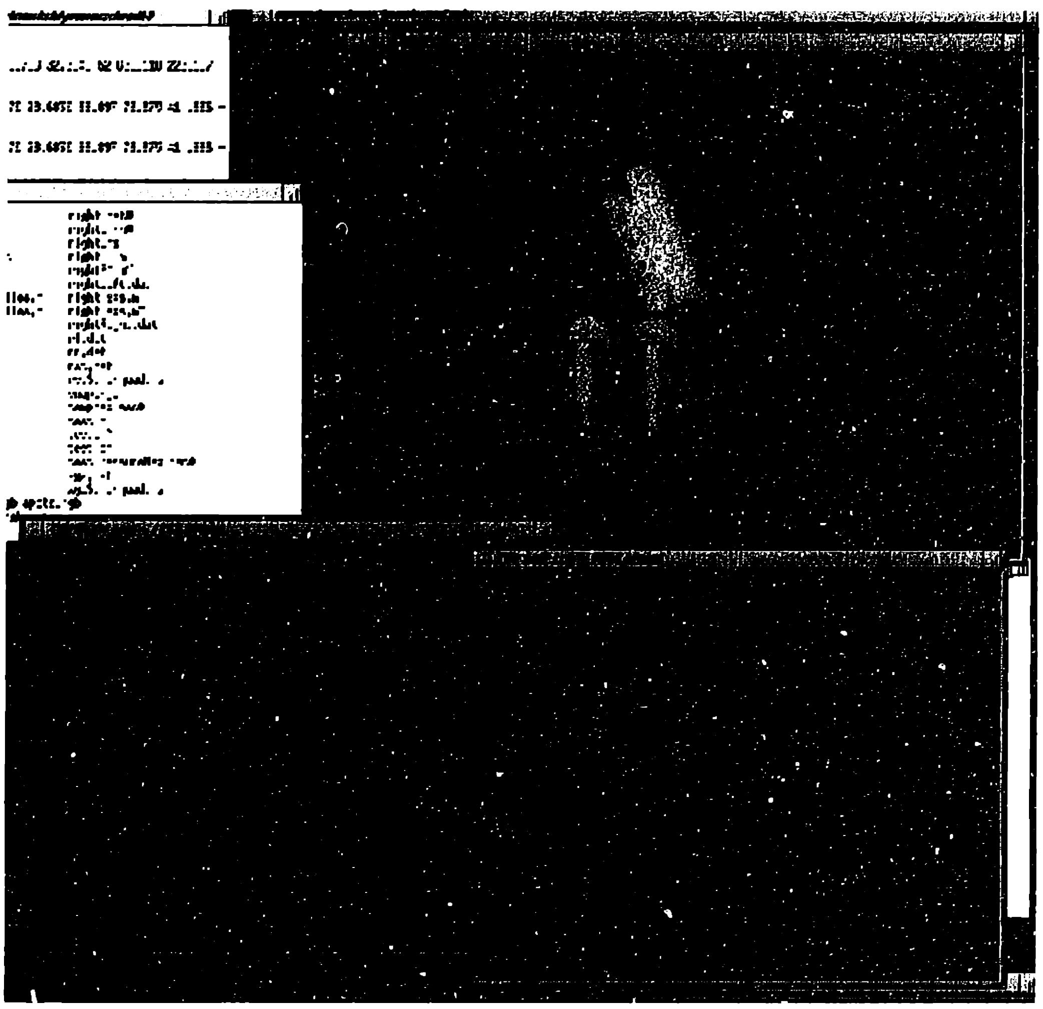
Our experimental setup utilized an Octane class workstation running all three components on the same computer simultaneously. Computer graphics visualization of the physical system is implemented using the SGI Open Inventor toolbox.

The SPoTS architecture consists of three main components: (See Figure 1.)

- The first component (represented by the left box in Figure 1) consists of the chair, the Tekscan pressure sensor system, and preprocessing code that segments out the pressure sub-maps for the left and the right thighs models.

- The second component (represented by the middle box in Figure 1) consists of a finite element solver that takes the pressure data from the first component and applies the forces to the FEM model of the left and right thighs. It calculates the reaction forces necessary to keep the knee and hip joints of the thigh models stationary. It also displays graphics depicting strain state of each thigh model.

- The third component (represented by the right box in Figure 1) consists of a constraint enforcing system that balances the reaction forces from the thigh models with the gravitational force exerted by the torso. It also displays a graphic depicting the leaning state.



[illegible]

Figure 2 - SPoTS running on a Silicon Graphics machine. The top window displays the leaning state of the body. The lower two windows depict a representation of the strain state of the right and left thighs.

These three components work in sequence to calculate the two leaning angles from the raw pressure data. (See Figure 2.)

SPoTS makes a couple assumptions that are crucial to the system. First, it assumes that the person is seated in the chair with both feet on the floor at all times. This assumption is made because in calculating the reaction forces in the FEM thigh models, we fixed the positions of the knee and hips in the model. We believe this assumption is acceptable because when one sits and leans in a chair with boot feet on the floor, the position of the knees and hips stay relatively fixed.

Another assumption that we make is that the seated person is in balance at all times, i.e., the person is not off balance or losing balance when the contact pressure distribution is read. We also assume that the person is not moving very fast, e.g., spasming left and right in the chair. These assumptions are made because we presume that we have a quasi-static situation where all dynamics at each frame of the simulation settle down before the next frame. If the person moves rapidly or is out of balance, dynamic effects at each frame would become important but would not be accounted for in the model.

2.2 Preprocessing using Expectation Maximization

The contact pressure data that the Tekscan sensor system outputs is similar to that shown in Figure 3. The top of the figure corresponds to the front of the body where the knees are located, and the bottom, to the backside of the body. The left and right sides of the figure are

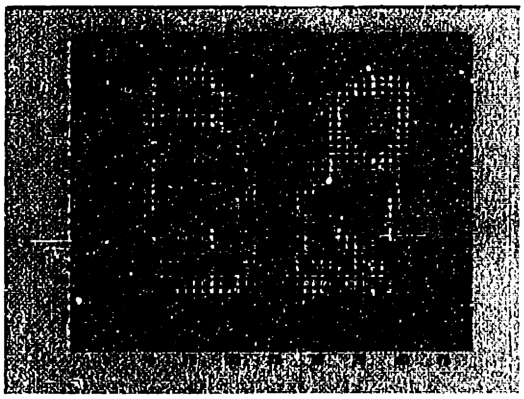


Figure 3 - Tekscan 42x48 pressure map.

reversed – the left side of the image corresponds to the right side of the chair, and vice versa. Thus, the right "blob" in the figure corresponds to the contact pressure from the left thigh, and the left "blob," the pressure from the right thigh. The coordinate system associated with the pressure map starts at (1,1) in the lower left corner of the figure and ends at (48,42) in the upper right corner.

In order to separate this pressure map into two smaller portions that we can map onto the surfaces of the two FEM thigh models, we use the expectation maximization algorithm (EM) as presented by Bishop [2] to "fit" a mixture of four gaussian densities to the data. We then use two of the four gaussians to recreate each pressure "blob" (the left blob and the right blob in Figure 3) that will be mapped to the FEM thigh models.

The weighted expectation maximization algorithm we use in SPoTS is an iterative method for finding a good fit for a mixture of our four target gaussians to our pressure data. Each gaussian is associated with five parameters that completely define the gaussian. These parameters are the mean in the x direction, μ_x, the mean in the y direction, μ_y, the variance in the x-direction, σ_{xx}, the covariance between x and y, σ_{xy}, and the variance in the y-direction, σ_{yy}. Graphically, the x and y means define the location of the center of the gaussian, and the covariance matrix (a 2x2 matrix consisting of the variances on the diagonal and the covariance on the off-diagonals) define the horizontal and vertical spread of the gaussian as well as the rotation of the gaussian.

The first step in the EM process is to guess the initial gaussian parameters. For the SPoTS preprocessor, we chose as the initial location for the four gaussians the four points located at (13,11), (13, 33), (37,11), (37,33). For the initial covariance matrix for the four gaussians, we initialized with the covariance matrix corresponding to all the data points combined.

In the EM algorithm, we define the data vector $\mathbf{x}^n$ to be a vector (x, y) corresponding to the position of the datapoint n. The superscript n ranges from 1 to the total number of data points in the pressure map. We also introduce the variable z^n, which is an integer in the range of 1 to 4 specifying which of the four gaussians generated the data point n. Also, E^{comp} is the negative log-likelihood (or error function) for the complete data problem:

$$E^{comp} = -\sum_{n=1}^{N} \ln\{P^{new}(z^n)p^{new}(\mathbf{x}^n|z^n)\} \tag{1}$$

where $P^{new}(z^n)$ is the probability of the gaussian, and $p^{new}(\mathbf{x}^n | z^n)$ is the probability of $\mathbf{x}^n$ given the gaussian. The pressure at each point is used as a weighting factor.

To find the probability distribution of the $\{z^n\}$, we can use our initial guesses and compute the expectation of E^{comp} with respect to this distribution. This corresponds to the expectation step of EM.

The next step in the EM algorithm is to find the "new" parameter values by minimizing the expected error and maximizing the likelihood with respect to these parameters. This corresponds to the maximization step of EM.

These new parameter values are then fed back as the initial guesses for the next iteration, and the process is repeated until the likelihood for the complete data problem is sufficiently maximized. With the data preprocessor in SPoTS, we iterate twenty times to get a good fitting of gaussian mixtures to the pressure data. (See Figure 4.)

From these gaussian parameters we can recreate a pressure map that is very similar to the original map. The two main differences between the original and the recovered pressure maps are:

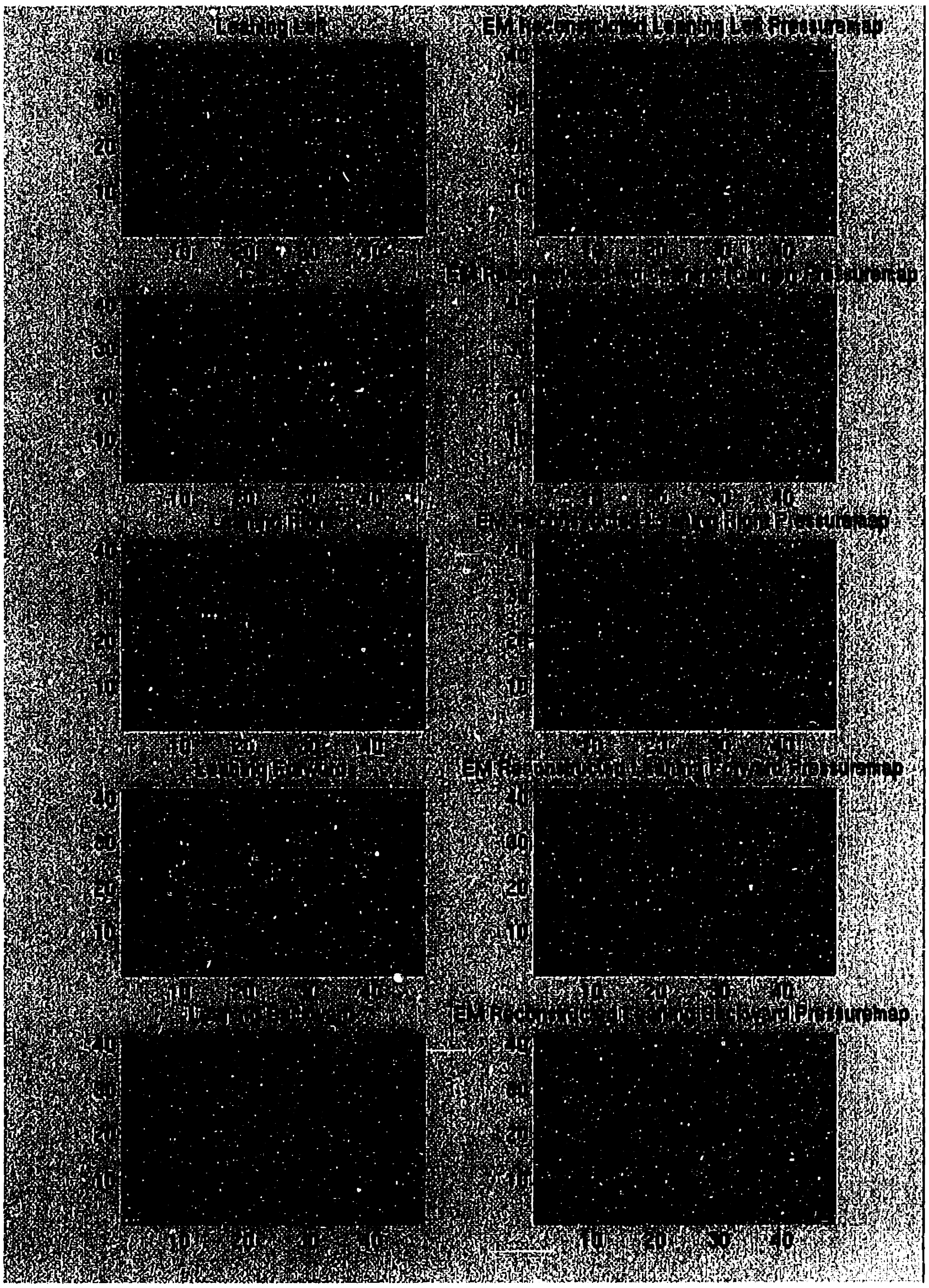


Figure 4 – The left figures are EM fitting of four gaussian distributions to pressure maps - the ellipses corresponding to the gaussians, two standard deviations from the mean. The right figures are the recovered pressure maps from gaussian parameters. The corresponding postures are leaning left 15 degrees, no leaning, right 15 degrees, forward 15 degrees, and backward 15 degrees.

(1) There is a scaling difference between the original and the reconstructed maps because the values in the recovered map represent probabilities and are thus all less than unity. We remedy this by scaling all the values in the reconstructed map so that the highest point in this map is the same as the highest point in the original map.

(2) The recovered map is the best fit of a mixture of four gaussians to the pressure data – there will be an automatic smoothing of data in the recovered pressure map. This regularization can be considered a feature of the system because it filters out noise in the pressure map.

We gain two very important benefits from performing EM on the data:

(1) We obtain a closed-form formulation of the pressure map. Given any point (x, y) on the map, we can compute a corresponding pressure by adding the contributions of each of the four gaussians at that point.

(2) We also gain the ability to segment out the two pressure sub-maps to apply to each of the FEM thigh models. Looking at Figure 4, we see that four gaussians cluster into two "blobs" each made up of two gaussians with very little overlap between the blobs. Although the entire pressure map is formed from a mixture of all four gaussians, since most of the data in the gaussians is contained within two standard deviations from the mean of each gaussian, we see that the gaussians in the left lobe do not

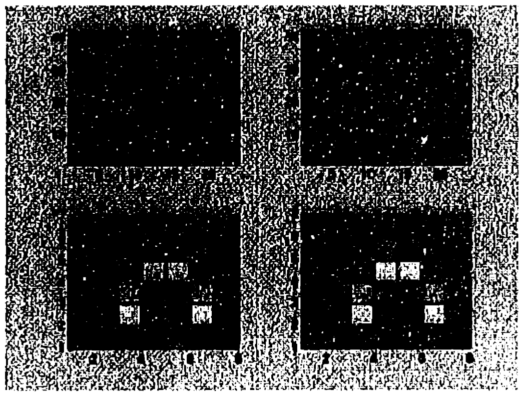


Figure 5 - Resampling of the pressure data.

add substantially to the points in the right lobe, and vice versa. Therefore, we can simply use a mixture of only two gaussians to construct the sub-maps to be mapped on each FEM thigh model.

The next step of the preprocessing procedure is to resize the pressure maps to dimensions compatible with the left and right FEM thigh models. We reduce the dimensions from a 42x24 map to a 7x8 map by locally averaging the pressure sub-maps. (See Figure 5.)

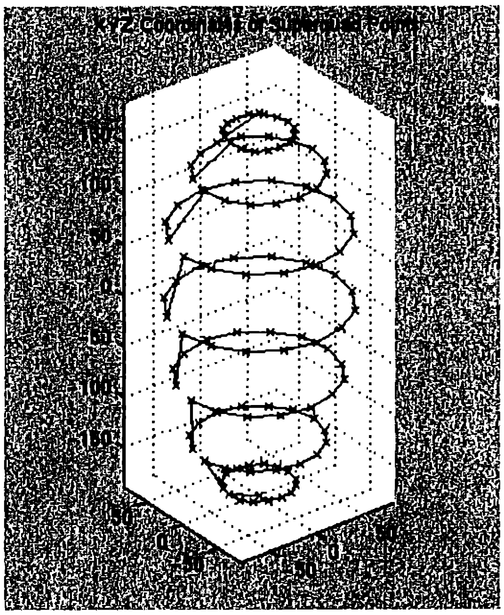


Figure 6 - Geometry of the thigh models.

We than map these pressure maps onto the thigh model surfaces by applying each pressure reading in the 7x8 sub-map to the points on the thigh model (See Figure 6.)

2.3 Finite Element Method and Thingworld

Thingworld [15] is physics simulation and animation system developed in the Vision and Modeling Group at the MIT Media Laboratory that uses the physics of a single 27-node isoparametric finite element to animate deformable graphic objects.

The system utilizes the finite element method (FEM), which is a numerical method for approximating the physics of a deformable object. The isoparametric element used is a 27-node three-dimensional element in a 3x3x3-cube configuration. (See Figure 5.)

Contrasted to the finite difference method, which computes the physics of the object only at the nodes, the finite element method also models the physics in between the nodes using a set of interpolation functions, **H**(r, s, t). The interpolation functions, also called shape functions, allow us to go between the undeformed local space (r, s, t) and deformed space (x, y, z):

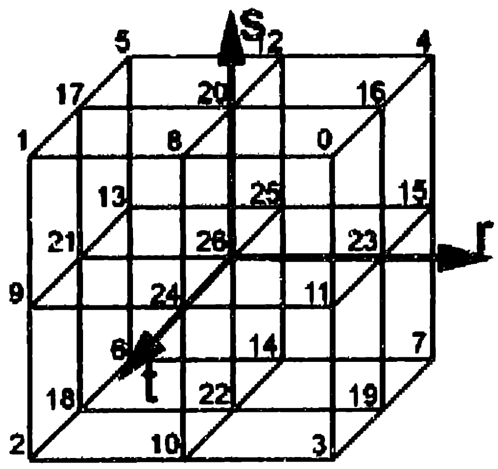


Figure 7 - 27-node isoparametric finite element.

$$\mathbf{u}(x,y,z) = \mathbf{H}(r,s,t)\hat{\mathbf{u}} \tag{2}$$

where $\hat{\mathbf{u}}$ represents the nodal displacements, **H** is the interpolation matrix of interpolation functions, and **u** is the displacement polynomial.

The strain ε, which describes the deformations caused by stresses exerted on the body, can be found by using the strain-displacement transformation matrix **B**:

$$\varepsilon = \mathbf{B}\hat{\mathbf{u}}. \tag{3}$$

B is obtained by differentiating **H** with respect to r, s, and t, and premultiplying the result by the inverse of the Jacobian operator.

How an elastic object deforms is associated with the object's stiffness matrix. To find the stiffness matrix **K** for the element, we integrate the following over the volume of the element:

$$\mathbf{K} = \int \mathbf{B}^T \mathbf{C} \mathbf{B} dV \tag{4}$$

where **C** is the material matrix describing the stress-strain relationship between the various degrees of freedom in the element. It should be noted that the volume integration extends over the natural coordinate volume, and in general, we need to make the following transformation:

$$dV = \det \mathbf{J}\ dr\ ds\ dt \tag{5}$$

where det **J** is the determinant of the Jacobian operator.

The material matrix **C** for a three-dimensional isoparametric finite element is

$$C = \frac{E(1-v)}{(1+v)(1-2v)} \begin{bmatrix} 1 & \frac{v}{1-v} & \frac{v}{1-v} & 0 & 0 & 0 \\ \frac{v}{1-v} & 1 & \frac{v}{1-v} & 0 & 0 & 0 \\ \frac{v}{1-v} & \frac{v}{1-v} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2v}{2(1-v)} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2v}{2(1-v)} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2v}{2(1-v)} \end{bmatrix} \tag{6}$$

where E is the Young's modulus and v is the Poisson's ratio.

In SPoTS, we assume quasi-statics, which means that at each time step in the simulation, the dynamic interplay between external forces, material elasticity and inertia equilibrate before the next time step. This simplifies the physics from the generalized equations of motion involving inertia and damping

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{C}\dot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{R} \tag{7}$$

to the quasi-static form

$$\mathbf{K}\mathbf{u} = \mathbf{R} \tag{8}$$

where **R** is the sum of the external body forces $\mathbf{R}_B$ and surface forces $\mathbf{R}_S$, and **u** represents the nodal displacements.

To get the nodal surface force matrix $\mathbf{R}_S$, we again turn to the interpolation functions:

$$\mathbf{R}_S = \int_S \mathbf{H}^{S^T} \mathbf{f}^S dS. \tag{9}$$

The vector $\mathbf{f}^S$ is the pressure sub-map that we constructed in the preprocessing step.

With the surface forces applied to the FEM thigh models, we can solve for the reaction forces at the knee and the hip of the FEM thigh models with the assumption these points remain fixed. This assumption corresponds to fixing nodes 24 and 25 in each FEM thigh model. (See Figure 7.) We believe that it is fair to make this assumption because the hip and the knees stay relatively fixed when someone is leaning.

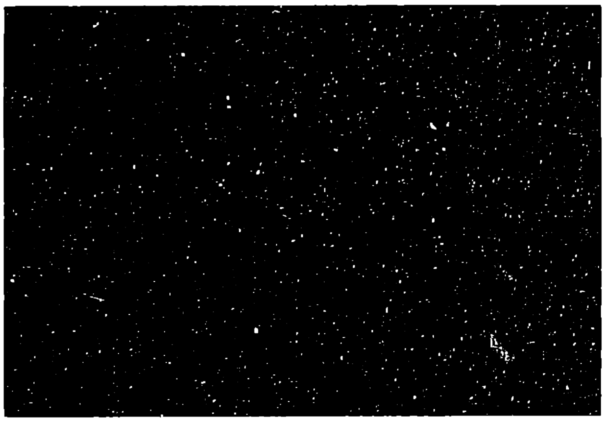


Figure 8 - Some modes of vibration of the thigh model.

We remove the rows and columns in the stiffness matrix corresponding to the fixed nodes and solve for the displacements of the other nodes that are free to move using gauss elimination. We can then back-substitute the displacements and calculate the reaction forces at nodes 24 and 25 necessary to keep them fixed.

Because the Thingworld system pre-computes the modes of vibration of the stiffness matrix (see Figure 8) and animates by superposing these modes, we convert from nodal displacements **u** to modal displacements **X** for so that Thingworld knows how to display the objects on the screen:

$$\mathbf{X} = \Phi^{T}\mathbf{M}\mathbf{u}. \tag{10}$$

2.4 A Modular Constraint System and BodyModel

BodyModel is an object-oriented modular constraint management and animation system implemented at the Vision and Modeling Group at the MIT Media Laboratory. [23] Solid physical objects with masses and rotational moments of inertia can be quickly connected together using hard constraints such as pin and ball joints or soft constraints like linear and rotational springs. We first solve for the correct leaning angles θ and φ by enforcing a force and torque balance, (see Figure 9) and then use the BodyModel system to visualize the corresponding posture.

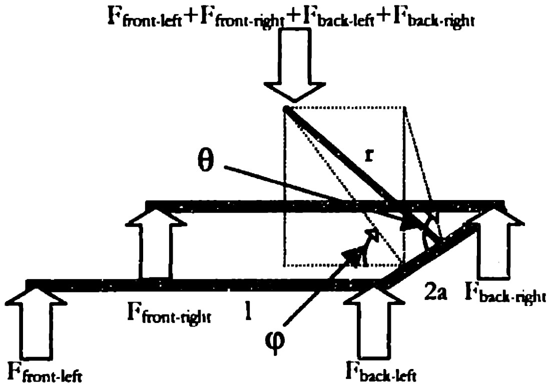


Figure 9 - Force balance.

We calculate θ, or how much the model is leaning to the left or right by the following:

$$\begin{aligned} \sum forces &= F_{left} + F_{right} - F_{body} = 0 \\ \sum torque &= F_{right}a + F_{body}r\cos\theta - F_{left}a = 0 \end{aligned} \tag{11}$$

where the lengths *r* and *a* are as shown in Figure 9, F_{left} is the sum of $F_{front\text{-}left}$ and $F_{back\text{-}left}$, and F_{right} is the sum of $F_{front\text{-}right}$ and $F_{back\text{-}right}$. These two equations combine to become

$$\theta = \cos^{-1}\left[\frac{(F_{left} - F_{right})}{(F_{left} + F_{right})}\frac{a}{r}\right]. \tag{12}$$

We balance the left and right forces to find φ:

$$\begin{aligned}\sum forces &= F_{frontt} + F_{back} - F_{body} = 0 \\ \sum torque &= F_{body} r \sin\theta \cos\varphi - F_{front} l = 0\end{aligned} \tag{12}$$

where the length *l* is shown in Figure 9, F_{front} is the sum of $F_{front\text{-}left}$ and $F_{front\text{-}right}$, and F_{back} is the sum of $F_{back\text{-}left}$ and $F_{back\text{-}right}$. These equations combine to form

$$\varphi = \cos^{-1}\left[\frac{F_{front}}{F_{front} + F_{back}}\frac{l}{r\sin\theta}\right]. \tag{13}$$

We use angles θ and φ to make the model lean correctly. (See Figure 10.)

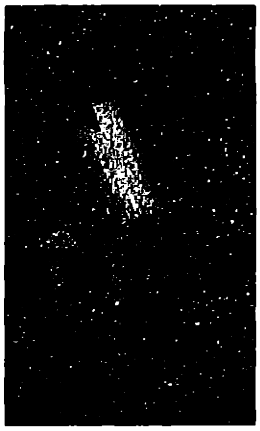


Figure 10 - The puppet leaning to the right.

Chapter 3

TESTING AND RESULTS

3.1 Training the System

The training for the system was performed by capturing five pressure maps corresponding to the five posture – no leaning, leaning left, leaning right, leaning forward, and leaning backward, all 15 degrees from vertical. The body angles were measured visually with a protractor.

The system was calibrated using these five control points. The intended theta and intended phi columns in Table 1 are what ideal training data would yield as the inclination angles for each posture. However, it was difficult, for example, to lean only to the left or right without leaning forward or backward. The theta and phi columns are the closest calculated posture angles after fitting the five control points.

Leaning Position	*Theta*	*Phi*	*Intended theta*	*Intended phi*
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Table 1 The calibration points for the system

3.2 Performance of the System

To test the system, we collected two additional sets of pressure data. For the both sets of test data, the subjects were asked to sit upright, lean left, lean right, lean forward, and lean backward at an angle 30 degrees from the vertical. The angle of inclination was measured visually using a protractor.

The first set of test data was collected from the same person as the training data was collected. He is a 5'6" male weighing 145 lbs. The results of running the data through the system are shown in Table 2. The system was running at approximately 8 frames per second for this test.

Leaning Position	Theta	Phi	Intended theta	Intended phi	% Error theta	% Error phi
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Table 2 – Calculated and intended posture angles for subject A.

The second set of data was collected from a second subject. He is a 5'10" male weighing 200 lbs. Data for the leaning backward posture was not used because the subject could not lean backward 30 degrees without hitting the backrest. The results of running his data through the system are shown in Table 3. The system was running at approximately 8 frames per second for the second test.

Leaning Position	Theta	Phi	Intended theta	Intended phi	% Error theta	% Error phi
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Table 2 – Calculated and intended posture angles for Subject B.

Chapter 4

DISCUSSION

Overall the performance of the system was varied. Although the system did predict posture angles that were as high as 34% off from the intended angle, it did determine qualitatively the direction that the person was leaning, i.e., if somebody leaned left. The system correctly determined when a person was leaning to the left or right, although there was some error in determining the amount of lean in these directions. The system was not as sensitive to leaning forward and backward.

We believe that the low performance by the system in predicting how much the person is leaning may be due to two major factors. The first is that there seems to be a lot of coupling between leaning left and right with leaning forward and backward – it was extremely difficult to constrain the leaning posture to just one degree of freedom. This factor is even more important when we realize that the coupling occurred in the training data where it was assumed to be nonexistent. Errors in the training would obviously lead to errors in tracking.

The second reason why the error rates were high may be because our assumption that the hips remain stationary may have broken down. This would lead to high error rates because one could generate different contact pressure distributions while still remaining in what looks like the same posture, and vice versa.

There is some evidence that the stationary hip assumption may be somewhat invalid. By simply looking at the pressure maps of sitting upright for a number of people, one sees that there is usually more pressure on one side of the body than on the other, despite the fact that they are all sitting upright. This phenomenon would have an affect on the accuracy of posture tracking.

The future direction of this work should be focused on determining the cause of error in the system and then compensating for the error. Perhaps by modeling the behavioral control of posture in addition to mere physical balance (people probably don't lean in such a way that they have to actively keep their balance) we could achieve better tracking accuracy. Perhaps instead using the fixed knee / fixed hip assumption and solving for the reaction forces at these joints we should model contact explicitly. These are all possible avenues of exploration that we should try in the future.

APPENDIX A: DATA PREPROCESSING CODE

The data preprocessing component of the SPoTS system is composed of expectation maximization routines coded in C++ and data matrix processing code written in Matlab script.

The expectation maximization code is located in /u/ilu/bigu/Bodymodel/serial on the MIT Media Laboratory file server. It includes the following files:

Makefile	The gmake makefile for creating the binaries.
buttview.c++	The main c++ file. It reads in a Tekscan data file either from a file or over the serial port and applies the estimation maximization routines to the data. It outputs the gaussian parameters to the screen.
eispack.h	A header file for the eispack single value decomposition library.
em.c++	The routines for the expectation maximization algorithm.
em.h	The header file for the expectation maximization routines.
lu_solve.c	The routines for the gauss elimination and backsubstitution algorithms.
lu_solve.h	The header file for the gauss elimination and backsubstitution algorithms.
matrix.c	The routines for matrix arithmetic.
matrix.h	The header file for the matrix arithmetic routines.
matrix_eispack.c	The routines that allow matrices access to the eispack library.

matrix_eispack.h	The header file for the routines that allow matrices access to the eispack library.
matrix_init.c	Routines for creating and destroying matrix structures.
matrix_init.h	The header file for the matrix creation and destruction routines.
scanargs.c	Routines for parsing the command line arguments.
scanargs.h	The header file for the command line argument parsing routines.
serial.c++	Routines for sending and receiving data over the serial port.
serial.h	The header file for the serial port routines.

The data matrix processing code is located in /u/ilu/prints on the MIT Media Laboratory file server. It includes the following files:

backwardEM.m	Recreates the backward pressure map from the gaussian parameters.
backwardMaps.m	Creates intermediate data sub-map files for the backward pressure map.
centerEM.m	Recreates the center pressure map from the gaussian parameters.
drawbackward.m	Plots the backward pressure map with the fitted gaussians superimposed on the graph.
drawcenter.m	Plots the center pressure map with the fitted gaussians superimposed on the graph.
drawforward.m	Plots the forward pressure map with the fitted gaussians superimposed on the graph.

drawleft.m	Plots the left pressure map with the fitted gaussians superimposed on the graph.
drawright.m	Plots the right pressure map with the fitted gaussians superimposed on the graph.
ellipse.m	Draws an ellipse given a center (x, y) and a covariance matrix.
forwardEM.m	Recreates the forward pressure map from the gaussian parameters.
forwardMaps.m	Creates intermediate data sub-map files for the forward pressure map.
leftEM.m	Recreates the left pressure map from the gaussian parameters.
leftMaps.m	Creates intermediate data sub-map files for the left pressure map.
makeAllDataFile s.m	Takes all the intermediate data sub-map files and makes the final sub-map files.
rightEM.m	Recreates the right pressure map from the gaussian parameters.
rightMaps.m	Creates intermediate data sub-map files for the right pressure map.

APPENDIX B: MODIFIED THINGWORLD CODE

The modified Thingworld component of the SPoTS system is a collection of routines located in /u/ilu/bigu/Bodymodel/modal-1.1/src on the MIT Media Laboratory file server. It includes the following files:

Makefile	The gmake file for creating the binaries.
applyForces.c++	The routines for applying surface forces to ThingWorld objects.
applyForces.h	The header file for surface force application routines.
butt-tw2.c++	The main source file for the component.
colormapLookup.c++	The routine for looking up a value in a colormap and returning the corresponding RGB color.
colormapLookup.h	The header file for the colormap lookup routine.
decode.c++	The routine for decoding data in encoded in base 128.
decode.h	The header file for the decoding routine.
displace.c	Routines for calculating the displacements in the superquad.
dmap30.c	Routines for calculating the displacement map for a 30 DOF system.
dmap81.c	Routines for calculating the displacement map for an 81 DOF system.
draw_gl.c	The draw routines for openGL.

encode.c++	The routines for encoding data in base 128.
encode.h	The header file for the encoding routine.
fem.c	Routines used to determine forward and inverse polynomial mappings from superquadric space to isoparametric finite element space, and from deformed space to superquadric space.
fem81.c	Routines for computing the value and the partials for FEM interpolation polynomial for a 27-node isoparametric element.
fit30.c	Routines for fitting data points in a 30 DOF system.
fit81.c	Routines for fitting data points in a 81 DOF system.
fit_ellipsoid.c	Routines for finding an initial superquad ellipsoid from data points.
fit_ellipsoid.h	The header file for the routines for finding the initial superquad.
ivDisplay.c++	Routines for drawing the superquad on the screen using Inventor.
ivDisplay.h	The header file for the Inventor drawing routines.
lsq.c	Routines for weighted least squares.
lu_solve.c	Routines for gauss elimination.
lu_solve.h	The header file for the gauss elimination routines.
mass.c	Routine for finding the mass matrix.
matrix.c	Matrix arithmetic routines.

matrix.h	The header file for the matrix arithmetic routines.
matrix_eispack.c	Routines for using the eispack library with matrices.
matrix_init.c	Routines for creating and destroying matrix structures.
matrix_init.h	The header file for matrix creation and destruction routines.
phi30.c	Routines for finding the modes of vibration for a 30 DOF system.
phi81.c	Routines for finding the modes of vibration for a 81 DOF system.
point_mapping.c	Routines that maps forces.
point_mapping.h	The header file for routines that map forces.
polygons.c	Routines for superquad sampling.
rotate_phi.c	Routine to rotate phi.
rotation.c	Routines for dealing with rotations.
scanargs.c	Routines for parsing the command line arguments.
scanargs.h	The header file for the command line argument parser routines.
sq.c	Routines for setting up the superquad.
sq.h	The header file for superquad initialization routines.
sq30.c	Routines to manipulate a 30 mode superquad.
sq81.c	Routines to manipulate an 81 mode superquad.

stiffness.c	Routines to create the stiffness matrix.
svd_matrix_invert.c	Routine that uses svd to solve a weighted least squares problem.
testIdle.c++	Routines that are performed while Inventor is idle.
testIdle.h	The header file for the routines performed during the Inventor idle callback.
undeform30.c	Routines to undeform a point back to the original space.
undeform81.c	Routines to undeform a point back to the original space.

APPENDIX C: BODYMODEL CODE

The BodyModel component of the SPoTS system is a collection of routines located in /u/ilu/bigu/Bodymodel/Constraint/dynamics/ on the MIT Media Laboratory file server. It includes the following files:

Client.c++	The main client object.
Client.h++	The header file for the main client object.
Connector.h++	The header file for connectors.
Connector2D.c++	2D connector objects.
Connector3D.c++	3D connector objects.
ConnectorQ.c++	Connector points on quat objects.
Constraint.c++	Constraints – ball and pin joints, etc.
Constraint.h++	The header file for constraint objects.
Integrator.c++	The numerical integrator.
Integrator.h++	The header file for the integrator.
Interface.h++	The interface object.
Makefile	The gmake file for creating the binaries.
Matrix.c++	The matrix class.
Matrix.h++	The header file for the matrix class.
Object.c++	Generic objects.

Object.h++	Header file for the generic objects.
QObject.c++	Quat objects.
QObject.h++	Header file for quat objects.
Quaternion.h++	Quaternion routines.
Server.c++	The graphics server object.
Server.h++	The header file for the graphics server.
Servo.c++	Linear and rotational spring objects.
Servo.h++	The header file for linear and rotational spring objects.
State.c++	Vector class.
State.h++	The header file for the vector class.
buttmodelPVM.c++	The main file.
scanargs.c	Routines for parsing command line arguments.
scanargs.h	The header file for the parser.

APPENDIX D: PARALLEL VIRTUAL MACHINE

Parallel virtual machine (PVM) is a software package that allows multiple computer processes running on the same or multiple computers to communicate with each other and trade information with minimum overhead. PVM is supported on a variety of computer platforms including Alpha, HP, Intel / Linux, Intel / Win32 and SGI. The official support web site is located at http://www.epm.ornl.gov/pvm/.

PVM is used in SPoTS to allow the modified Thingworld component to talk to the BodyModel component. All that is necessary to use PVM is to include the header file pvm3.h in the source code, and to link to the two PVM libraries – libpvm3.a and libgpvm3.a – during compilation of the executable.

Everything is explained in detail in the online manuals at the official web site. Also, extensive documentation as well as the libraries themselves are available on the MIT Media Laboratory file server at /mas/vision/built/pvm3.

BIBLIOGRAPHY

[1] Sumit Basu. *A Three-Dimensional Model of Human Lip Motion*. MS thesis, Massachusetts Institute of Technology, 1997.

[2] Christopher M. Bishop, *Neural Networks for Pattern Recognition*. Oxford University Press, New York, NY, 1995.

[3] Klaus-Jürgen Bathe, *Finite Element Methods*. Prentice Hall, Englewood Cliffs, NJ, 1996.

[4] Terrance E. Boult, Samuel D. Fenster, and Thomas O'Donnell. "Physics in a Fantasy World vs. Robust Statistical Estimation". In *Proceedings of the NSF/ARPA Workshop on 3-D Object Representation for Computer vision*, 1994.

[5] Kiran Dandekar. *Role of Mechanics in Tactile Sensing of Shape*. PhD thesis, Massachusetts Institute of Technology, 1995.

[6] Irfan A. Essa, Stan Sclaroff, and Alex Pentland. "Physically-based Modeling for Graphics and Vision". In *Directions in Geometric Computing*. Ralph Martin, Editor. Information Geometers, Great Britain, 1993.

[7] Y.C. Fung. *Biomechanics: Mechanical Properties of Living Tissue*. Springer-Verlag, New York, 1993.

[8] Herbert Goldstein. *Classical Mechanics*, 2nd edition. Addison-Wesley, Reading, MA, 1980.

[9] David Halliday, and Robert Resnick. *Fundamentals of Physics*. John Wiley and Sons, New York, NY, 1988.

[10] K.L. Johnson. *Contact Mechanics*. Cambridge University Press, Great Britain, 1985.

[11] Al Kelley and Ira Pohl. *A Book on C*, 2nd edition. Benjamin/Cummings Publishing, Reading, MA, 1995.

[12] Stanley B. Lippman. *C++ Primer*, 2nd edition. Addison-Wesley Publishing, Reading, MA, 1991.

[13] Dianne Tanya Victoria Pawluk. *A Viscoelastic Model of the Human Fingerpad and a Holistic Model of Human Touch*, PhD thesis, Harvard University, 1997.

[14] Alex Pentland and Stan Sclaroff. "Closed-Form Solutions for Physically Based Shape Modeling and Recognition". In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 13, Number 7, July 1991.

[15] Alex Pentland and John Williams. "Good Vibrations: Modal Dynamics for Graphics and Animation". In *Computer Graphics*, Volume 23, Number 3, July 1989.

[16] Stan Sclaroff and Alex Pentland. "Generalized Implicit Functions for Computer Graphics". In *Computer Graphics*, Volume 25, Number 4, July 1991.

[17] Gilbert Strang, *Introduction to Applied Mathematics*, Wellesley-Cambridge Press, Wellesley, MA, 1986.

[18] Hong Z. Tan, Ifung Lu and Alex Pentland. "The Chair as a Novel Haptic User Interface." In *Proceedings of the Workshop on Perceptual User Interfaces*, 1997.

[19] Matthew Turk and Alex Pentland. "Eigenfaces for Recognition". In *Journal of Cognitive Neuroscience*, Volume 3, Number 1, pages 71-86, 1991.

[20] Josie Wernecke. *The Inventor Mentor*. Addison-Wesley Publishing, Wellesley, MA, 1994.

[21] John R. Williams and Alex P. Pentland. Superquadratic Object Representation for Dynamics of Multi-body Structures. MIT Media Laboratory Technical Report No. 110, MIT Media Lab Vision and Modeling Group, 1988.

[22] Andrew Witkin, Michael Gleicher, and William Welch. "Interactive Dynamics". In *Computer Graphics*, Volume 24, Number 2, March 1990.

[23] Christopher R. Wren and Alex Pentland. Dynamic Modeling of Human Motion. Technical Report No. 415, MIT Media Lab Vision and Modeling Group, 1997. Presented at I.U. Workshop, New Orleans, 1997.