

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/37990874>

A shoe-integrated sensor system for wireless gait analysis and real-time therapeutic feedback

Article

Source: OAI

CITATIONS

37

READS

138

2 authors, including:



[Stacy Bamberg](#)

University of Utah

38 PUBLICATIONS 453 CITATIONS

[SEE PROFILE](#)

**A Shoe-Integrated Sensor System for Wireless Gait Analysis
and Real-Time Therapeutic Feedback**

by

Stacy J. Morris

**M.S., Mechanical Engineering,
Massachusetts Institute of Technology, 1999**

**B.S., Mechanical Engineering,
Massachusetts Institute of Technology, 1996**

**SUBMITTED TO THE HARVARD-MIT DIVISION OF HEALTH SCIENCES AND
TECHNOLOGY IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE
DEGREE OF**

**DOCTOR OF SCIENCE IN MEDICAL ENGINEERING
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY**

JUNE 2004

**© 2004 Massachusetts Institute of Technology
All rights reserved.**

Signature of Author
Harvard-MIT Division of Health Sciences and Technology
May 10, 2004

Certified by
Joseph A. Paradiso, PhD
Associate Professor
Sony Career Development Professor of Media Arts and Sciences
MIT Media Laboratory
Thesis Supervisor

Accepted by
Martha L. Gray, Ph.D.
Edward Hood Taplin Professor of Medical and Electrical Engineering
Co-Director Harvard-MIT Division of Health Sciences and Technology

A Shoe-Integrated Sensor System for Wireless Gait Analysis and Real-Time Therapeutic Feedback

by

Stacy J. Morris

Submitted to the Harvard-MIT Division of Health Sciences and Technology
on May 10, 2004 in Partial Fulfillment of the
Requirements for the Degree of Doctor of Science
at the Massachusetts Institute of Technology

ABSTRACT

Clinical gait analysis currently involves either an expensive analysis in a motion laboratory, using highly accurate, if cumbersome, kinematic systems, or a qualitative analysis with a physician or physical therapist making visual observations. There is a need for a low cost device that falls in between these two methods, and can provide quantitative and repeatable results. In addition, continuous monitoring of gait would be useful for real-time physical rehabilitation.

To free patients from the confines of a motion laboratory, this thesis has resulted in a wireless wearable system capable of measuring many parameters relevant to gait analysis. The extensive sensor suite includes three orthogonal accelerometers, and three orthogonal gyroscopes, four force sensors, two bi-directional bend sensors, two dynamic pressure sensors, as well as electric field height sensors. The "GaitShoe" was built to be worn on any shoes, without interfering with gait, and was designed to collect data unobtrusively, in any environment, and over long periods of time.

Subject testing of the GaitShoe was carried out on ten healthy subjects with normal gait and five subjects with Parkinson's disease. The calibrated sensor outputs were analyzed, and compared to results obtained simultaneously from The Massachusetts General Hospital Biomotion Lab; the GaitShoe proved highly capable of detecting heel strike and toe off, as well as estimating orientation and position of the subject. A wide variety of features were developed from the calibrated sensor outputs, for use with standard pattern recognition techniques to classify the gait of the subject. The results of the classification demonstrated the ability of the GaitShoe to identify the subjects with Parkinson's disease, as well as individual subjects. Real-time feedback methods were developed to investigate the feasibility of using the continuous monitoring of gait for physical therapy and rehabilitation.

Thesis Supervisor: Prof. Joseph A. Paradiso
Associate Professor of Media Arts and Sciences

A Shoe-Integrated Sensor System for Wireless Gait Analysis and Real-Time Therapeutic Feedback

by

Stacy J. Morris

The following people served as committee members for this thesis:

..... Neville Hogan, Ph.D.

Professor of Mechanical Engineering
Professor of Brain and Cognitive Sciences
Massachusetts Institute of Technology

..... David E. Krebs, D.P.T., Ph.D.

Professor, MGH Institute of Health Professionals
Director, The Massachusetts General Hospital Biomotion Laboratory

..... Rosalind W. Picard, Sc.D.

Associate Professor of Media Arts and Sciences
MIT Media Laboratory

ACKNOWLEDGMENTS

To the Whitaker Foundation for supporting my education and research, to CIMIT for supporting this research, and to the Media Lab for additional support.

To Mike Harty and Thomas Papakosta at Tekscan, Inc., for sharing data about pressures in the insole.

To Jack Memishian at Analog Devices, for generously providing the demo yaw gyroscopes.

To Donna Scarborough for her enthusiasm and guidance of the clinical testing, to Dov Goldvasser for his help in the subject testing, and to the rest of the Biomotion Lab for their assistance.

To Dr. Leslie Shinobu for her enthusiasm and ideas for the use of the GaitShoe for persons with Parkinson's disease, and to her and her colleagues in the MGH Neurology Department, for helping to recruit volunteers with Parkinson's disease.

To the many volunteers for subject testing at the gait lab, for making this thesis possible.

To Dan Lovell for his tireless work on the ultrasound sensor, and to Erik Asmussen for his creativity in implementing the real-time feedback.

To Egon Pasztor for generously sharing his C code for the stripchart program.

To Ari Benbasat, for teaching me most of everything I know about electronics, microcontrollers, and debugging, for the initial work on the "stack", for all the great times in our office, and for his friendship.

To Hong Ma for his help and smiles, and to the rest of the Responsive Environments Group, for all the fun times. I look forward to seeing what you all do next!

To Asha Balakrishnan for her help in continuously finding yet another piece of equipment useful for calibration, and to her, Heather Gunter, and Amy Kerdok, for their friendship and support.

To the HST/MEMP program for providing the framework to make this research possible, and for a most interesting education.

To my wonderful committee: thank you, Neville Hogan, Dave Krebs, and Roz Picard for all your advice, brainstorming, suggestions, and ideas.

To Joe Paradiso, for bringing me into the group and the Media Lab, for opening my eyes everyday with his energy, curiosity, and never ending creativity for new technology and applications. I am honored to have done my doctorate research with you.

To MIT, for being my home for the last twelve years, and for being everything I could have dreamed of wanting for my education and more.

To my parents for their eternal support and encouragement.

And, to Ebbe for being my rock.

TABLE OF CONTENTS

Abstract	3
Acknowledgments	7
List of Illustrations	13
List of Tables	19
Chapter 1. Introduction	23
1.1 Thesis Statement	24
1.2 Motivation	24
1.3 Project Description	25
1.4 Important Gait Parameters	26
1.5 Summary of Contributions	27
Chapter 2. Background	29
2.1 Prior Work	29
2.1.1 On-Shoe Research Systems	29
2.1.2 Off-Shoe Approaches	34
2.1.3 Gait Recognition Systems	34
2.1.4 Commercial Systems	35
2.1.5 Expressive Footware: Instrumented Insole and Multiple Shoe-Based Sensors	36
2.2 Current State of the Art	38
2.2.1 Observational Gait Analysis	38
2.2.2 Overview of Clinical Gait Analysis	39
2.2.3 Optoelectronic Systems	40
2.2.4 Videographic Systems	41
2.2.5 Electromagnetic Systems	42
2.3 Clinical Need	43
2.3.1 Utility of On-Shoe Device in Subjects with Parkinson's Disease	44
2.3.2 Studying Gait Outside of the Motion Lab	46
Chapter 3. Hardware Design	49
3.1 Sensor Selection	49
3.2 Physical Implementation	52
3.2.1 The Stack	54
3.2.2 Insole Design	58
3.2.3 Shoe Attachment	59
3.2.4 The Basestation	62

3.3	Sensor Specifications	63
3.3.1	Accelerometer	63
3.3.2	Gyroscopes	67
3.3.3	Force Sensitive Resistors	71
3.3.4	Bend Sensors	75
3.3.5	Polyvinylidene Fluoride Strips	77
3.3.6	Electric Field Sensor	79
3.3.7	Ultrasound Sensor	85
3.4	Additional Components	86
3.4.1	Microcontroller	86
3.4.2	Radio Frequency Transceiver	88
3.4.3	Power	90
3.4.4	Other Components	93
3.5	The GaitShoe System	94
Chapter 4.	Sensor Analysis	97
4.1	Data Processing	98
4.1.1	Truncation	98
4.1.2	Time Adjustment	100
4.1.3	Data Adjustment	100
4.1.4	Calibration and Analysis	105
4.2	Analysis Model	107
4.2.1	Coordinate Systems	107
4.2.2	Inertial Measurement Evaluation	108
4.3	Gyroscopes	113
4.3.1	Calibration	113
4.3.2	Analysis of the Pitch	116
4.4	Accelerometers	122
4.4.1	Calibration	122
4.4.2	Orientation	125
4.4.3	Velocity and Stride Length Analysis	126
4.5	Force Sensitive Resistors	134
4.5.1	Calibration	134
4.6	Bend Sensors	141
4.6.1	Calibration	141
4.6.2	Placement of the Bend Sensors	146
4.6.3	Plantar Flexion and Dorsiflexion	147
4.7	PVDF Strips	149
4.8	Electric Field Sensor	153
4.9	Heel Strike and Toe Off Timing	156
Chapter 5.	Gait Parameter Validation	163
5.1	Pitch	164
5.2	Distance	166

5.3	Heel Strike and Toe Off	169
5.4	Discussion	170
Chapter 6.	Pattern Recognition Analysis	175
6.1	Pattern Recognition Techniques	175
6.1.1	Classification and Regression Trees (CART)	175
6.1.2	Bayes Decision Theory & Naïve Bayes	177
6.1.3	Support Vector Machines (SVM)	179
6.1.4	Neural Networks	181
6.2	Data Sets and Classifications	184
6.2.1	Gait Variation	185
6.2.2	Subjects and Data Sets	186
6.3	Feature Set	187
6.4	Hypotheses	192
6.5	Training and Testing Groups	193
6.6	Results	195
6.6.1	Hypothesis 1	196
6.6.2	Hypothesis 2	203
6.6.3	Hypothesis 3	205
6.6.4	CART Feature Information	206
6.6.5	Additional Neural Net Studies	209
6.7	Discussion	212
Chapter 7.	Real-Time Therapeutic Feedback	215
7.1	Overview	215
7.2	Rhythmic Auditory Stimulator	216
7.2.1	The RAS system	217
7.2.2	RAS Feedback	218
7.3	Conclusions	220
Chapter 8.	Conclusion	221
Appendix A.	Medical Information	227
A.1	Terminology	227
A.2	Parkinson's Disease	228
Appendix B.	Subject Testing	231
B.1	Study Design	231
B.2	Subject Information	236
Appendix C.	Pattern Recognition Information	239
C.1	Terminology	239
C.2	Complete Results	241

Appendix D. Hardware Information	255
D.1 Schematics and Board Layouts	255
D.2 Component Information	269
D.3 Other Hardware Information	271
Appendix E. Ultrasound Sensor	277
E.1 Circuit Boards	277
E.2 Microcontroller Code	281
E.3 Initial Results	283
Appendix F. Code	285
F.1 Microcontroller Code	285
F.2 Matlab Code	289
References	307

LIST OF ILLUSTRATIONS

Figure 2.1	Expressive Footware and its sensor outputs	37
Figure 2.2	A subject at the MGH Biomotion Lab	41
Figure 3.1	Schematic of the GaitShoe system	54
Figure 3.2	The stack	55
Figure 3.3	Photo of the front and back of the Main board	55
Figure 3.4	Photo of the front and back of the IMU board	56
Figure 3.5	Photo of the front and back of the Tactile board	56
Figure 3.6	Photos of the front and back of the transmit Ultrasound board for the right foot (right photos), and the front and back of the receive Ultrasound board for the left foot (left photos)	57
Figure 3.7	Photo of the front and back of the Power board	58
Figure 3.8	Photograph of an insole sensor	58
Figure 3.9	Bend sensor in ankle strap	59
Figure 3.10	Cross section showing the electric field sensor components	59
Figure 3.11	Photos of three versions of the shoe attachment	60
Figure 3.12	Final prototype of the GaitShoe attachment	61
Figure 3.13	Photo of the GaitShoe system on two shoes	61
Figure 3.14	Accelerometer response to impact at heel	62
Figure 3.15	Photo of the basestation	62
Figure 3.16	Schematic of the ADXL202E	65
Figure 3.17	Photo of the IMU board, with the ADXL202E components highlighted	65
Figure 3.18	Schematic of the ENC-03J	68
Figure 3.19	Photo of the IMU board, with the Murata ENC-03J components highlighted	68
Figure 3.20	Schematic of the ADXRS150	70
Figure 3.21	Photo of the IMU board, with the ADXRS150 components highlighted	70
Figure 3.22	Schematic of the FSR	74
Figure 3.23	Photo of the Tactile board, with the FSR components highlighted	74
Figure 3.24	Bend sensors shown back to back	75
Figure 3.25	Schematic of the bi-directional bend sensor	76
Figure 3.26	Photo of the Tactile board, with the bi-directional bend sensor components highlighted	77
Figure 3.27	Schematic of the PVDF strips	78
Figure 3.28	Photo of the Tactile board, with the PVDF strip components highlighted	78
Figure 3.29	Schematic showing electrode coupling to both the ground and the floor	80
Figure 3.30	Schematic showing the second electrode configuration coupling to the floor	81

Figure 3.31 Schematic of the electric field sensor	82
Figure 3.32 Photo of the Tactile board, with the electric field components highlighted	82
Figure 3.33 Example of an electrode set up on the bottom of a shoe to measure heel height	83
Figure 3.34 Example of two electrode set up on the bottom of a shoe to measure heel and toe height.	83
Figure 3.35 Ultrasound sensor for distance and angle between two shoes	85
Figure 3.36 Schematic of the Cygnal C8051F206	87
Figure 3.37 Photo of the main board, with the Cygnal C8051F206 components highlighted	87
Figure 3.38 Schematic of the RF Monolithics DR3000-1	89
Figure 3.39 Photo of the Main board, with the components of the RF Monolithics DR3000-1 highlighted.	89
Figure 3.40 Schematic of the Power Board	91
Figure 3.41 Header interconnection net names	94
Figure 3.42 GaitShoe hardware	94
Figure 3.43 High level block diagram of the GaitShoe system	95
Figure 3.44 Sensor outputs across all sensors for both feet	95
Figure 4.1 Complete raw output	98
Figure 4.2 Truncated section of the raw output	99
Figure 4.3 Outlier identified in bend sensor output	103
Figure 4.4 Outlier identified in accelerometer output	104
Figure 4.5 Bend sensor data with a number of adjustments	105
Figure 4.6 Frames of reference used in evaluation	107
Figure 4.7 The gait cycle	110
Figure 4.8 Comparison of IMU outputs during walking gait	111
Figure 4.9 Illustration for linear integration	112
Figure 4.10 Photo of the turntable used for the gyroscope calibration	114
Figure 4.11 Line fits for determining the sensitivity of the gyroscopes (shown for IMU-1)	115
Figure 4.12 Sample of direct linear integration of the z-gyroscope	117
Figure 4.13 Z-gyroscope output where angular velocity is non-zero during stance	117
Figure 4.14 Sample of linear integration of the z-gyroscope, with iteration	118
Figure 4.15 Sample of linear integration of the z-gyroscope, with iteration and spline-fit	119
Figure 4.16 Pitch of the foot during the gait cycle	120
Figure 4.17 Sample of linear integration of the z-gyroscope, with iteration and post-integration spline-fit	121
Figure 4.18 Final results, compared to BML data	121
Figure 4.19 Comparison of BML pitch to GaitShoe pitch	122
Figure 4.20 Demonstration of accelerometer calibration	123
Figure 4.21 Sample accelerometer calibration for determining sensitivity and zero-offset.	124
Figure 4.22 Determination of the orientation of the accelerometers	126

Figure 4.23	Dynamic acceleration along Xroom only	127
Figure 4.24	Dynamic acceleration with both Xroom and Yroom components	128
Figure 4.25	Integration of the acceleration in Xroom and Xshoe	131
Figure 4.26	Comparison of BML Xroom-displacement to GaitShoe Xroom- and Xshoe- displacement	132
Figure 4.27	Integration of the acceleration in Yroom	133
Figure 4.28	Comparison of BML Yroom-displacement to GaitShoe Yroom-displacement	133
Figure 4.29	Force applicators (above) and FSRs (below)	136
Figure 4.30	Test set-up for calibration of FSRs	137
Figure 4.31	Sample FSR-402 calibration data	137
Figure 4.32	Various line-fits to the FSR-402 calibration data	138
Figure 4.33	FSR-402 calibration curve	139
Figure 4.34	FSR-400 calibration curve	140
Figure 4.35	95% confidence intervals for FSR sensitivity curves	141
Figure 4.36	Method of calibration of bend sensors	142
Figure 4.37	Data from calibration of sensitivity of the bend sensor	143
Figure 4.38	Line fit for determining the sensitivity of bend sensors	144
Figure 4.39	Bend calibration routine, on a new sensor with and without tape	145
Figure 4.40	Bend sensor, calibrated with different pivot points	147
Figure 4.41	Plantar Flexion and Dorsiflexion from the GaitShoe (left) and the BML (right)	148
Figure 4.42	Calibrated insole bend sensor output	149
Figure 4.43	Various PVDF output plots	150
Figure 4.44	Reebok running shoe, Ecco walking shoe, and Dansko clog.	151
Figure 4.45	Comparison of PVDF output in three different types of shoes	151
Figure 4.46	Comparison of PVDF response to FSRsum response	152
Figure 4.47	Various electric field sensor output plots	154
Figure 4.48	Comparison of the electric field sensor output to foot height.	155
Figure 4.49	Sample output from the second electrode design	156
Figure 4.50	Comparison of the sum of the four calibrated [N] FSR outputs to the force plate output	157
Figure 4.51	Comparison of the sum of the four calibrated FSR outputs in pressure units of psi.	158
Figure 4.52	Determination of heel strike and toe off from FSRsum	159
Figure 4.53	First difference of spline-fit, zoomed-in	160
Figure 5.1	Comparison of GaitShoe pitch to BML foot array pitch	164
Figure 5.2	Histograms of the pitch validation results	166
Figure 5.3	Histograms of the stride length validation results	168
Figure 5.4	Histograms of the vertical displacement validation results	168
Figure 5.5	Histograms of the heel strike time and toe off time validation results	170

Figure 6.1	Examples of a single input neuron and a multiple input neural layer	182
Figure 6.2	Plots of maximum and minimum pitch, by age	191
Figure 6.3	Subject data of two most informative features in CART analyses	207
Figure 6.4	Subject data of two most informative features in CART analyses, individual PD subjects	208
Figure 7.1	Erik Asmussen and the Rhythmic Auditory Stimulator	216
Figure 7.2	Screenshots of the RAS system, showing the main menu (upper left), therapy configuration menu (upper right), and feedback control menu (lower).	217
Figure A.1	The Gait Cycle	228
Figure B.1	MGH consent form for subjects with healthy gait	233
Figure B.2	MGH consent form for subjects with difficulty walking	234
Figure B.3	MIT COUHES Consent Form	235
Figure B.4	Subject Testing Protocol, page 1 (left) and page 2 (right)	235
Figure C.1	Sample CART tree	239
Figure D.1	Schematic of the IMU board	256
Figure D.2	Layout of the top side of the IMU board	257
Figure D.3	Layout of the bottom side of the IMU board	258
Figure D.4	Pin mappings for the commercially available ADXRS150	259
Figure D.5	Schematic of the Tactile board	260
Figure D.6	Layout of the top side of the Tactile board	261
Figure D.7	Layout of the bottom side of the Tactile board	262
Figure D.8	Schematic of the Main board	263
Figure D.9	Layout of the top side of the Main board	264
Figure D.10	Layout of the bottom side of the Main board	265
Figure D.11	Schematic of the Power board	266
Figure D.12	Layout of the top side of the Power board	267
Figure D.13	Layout of the bottom side of the Power board	267
Figure D.14	Schematic of the Programming board	268
Figure D.15	Layout for the Programming board	268
Figure D.16	Component information, part one.	269
Figure D.17	Component information, part two	270
Figure D.18	Bend sensors soldered to wire	271
Figure D.19	Bend sensors with a "gob" of hot glue	271
Figure D.20	Hot glue flattened by fingers	272
Figure D.21	Trimmed hot glue on bend sensor solder connection	272
Figure D.22	Sketch of the pattern for both the left and right GaitShoe attachments	273
Figure D.23	Pattern traced onto PTG for forming the GaitShoe Attachment	273
Figure D.24	PTG cut to shape with heavy duty shears	274

Figure D.25 Heat gun and wooden block used to shape the PTG	274
Figure D.26 Sketch of the pattern for the battery enclosure	275
Figure D.27 GaitShoe attachment and battery enclosure; viewed separately from the back (left), and viewed together from the front (right)	275
Figure E.1 Photo of ultrasound hardware mounted on the GaitShoe hardware	278
Figure E.2 Schematic of the ultrasound transmit board.	278
Figure E.3 Schematic of the ultrasound transmit daughter board	279
Figure E.4 Schematic of the ultrasound receive board.	279
Figure E.5 Schematic of the ultrasound receive daughter board.	279
Figure E.6 Microcontroller code for the ultrasound transmit board.	281
Figure E.7 Microcontroller code for the ultrasound receiver board.	282
Figure E.8 Sample data from the two ultrasound sensors	283
Figure E.9 Data collected from all sensors, during slow gait	284
Figure F.1 206.h Code	285
Figure F.2 Basestation microcontroller code	286
Figure F.3 Stack microcontroller code, part one	287
Figure F.4 Stack microcontroller code, part 2	288
Figure F.5 Timing issues, as controlled by the basestation	289

LIST OF TABLES

TABLE 3.1	Sensor selection	52
TABLE 3.2	Relevant parameters of the Analog Devices ADXL202E accelerometer [71]	66
TABLE 3.3	Relevant parameters of the Murata ENC-03J gyroscope [73]	69
TABLE 3.4	Relevant parameters of the Analog Devices ADXRS150 gyroscope [75]	72
TABLE 3.5	Relevant parameters of the Interlink FSR-400 and FSR-402 [77]	74
TABLE 3.6	Relevant parameters of The Images Co. FLX-01 [78]	77
TABLE 3.7	Relevant parameters of the Measurement Specialties LDT0 PVDF strip [80]	79
TABLE 3.8	Relevant parameters of the Motorola MC33794DH [82]	84
TABLE 4.1	Final sensor order	101
TABLE 4.2	Order of calibrated and analyzed data	106
TABLE 4.3	Gyroscope sensitivities and zero offsets	115
TABLE 4.4	Accelerometer sensitivities and zero offsets	125
TABLE 4.5	Bend sensor sensitivities and zero offsets	145
TABLE 5.1	Pitch Validation Results	165
TABLE 5.2	Displacement Validation Results	167
TABLE 5.3	Heel Strike and Toe Off Validation Results	169
TABLE 6.1	Gender differences in gait [105]	185
TABLE 6.2	Summary of subject groupings	186
TABLE 6.3	Classifications of the ten female subjects	187
TABLE 6.4	Feature set	189
TABLE 6.5	Measurement exclusion due to dropped data	192
TABLE 6.6	Training and testing groups	193
TABLE 6.7	CART Hypothesis 1 results, using modified leave one out	196
TABLE 6.8	Naïve Bayes Hypothesis 1 results, using modified leave one out	197
TABLE 6.9	Neural Net Hypothesis 1 results, using modified leave one out	197
TABLE 6.10	CART Hypothesis 1 results, by class, using leave one subject out	198
TABLE 6.11	CART Hypothesis 1 results, by subject, using leave one subject out	198
TABLE 6.12	CART Hypothesis 1 results, using "free gait" as the test set	199
TABLE 6.13	CART Hypothesis 1 results, using "distracted gait" as the test set	199
TABLE 6.14	CART Hypothesis 1 results, using "paced gait" as the test set	199
TABLE 6.15	SVM Hypothesis 1.1 results, using modified leave one out	200
TABLE 6.16	SVM Hypothesis 1.2 results, using modified leave one out	200
TABLE 6.17	SVM Hypothesis 1.3 results, using modified leave one out	200

TABLE 6.18	SVM Hypothesis 1.1 results, using leave one subject out	201
TABLE 6.19	SVM Hypothesis 1.2 results, using leave one subject out	201
TABLE 6.20	SVM Hypothesis 1.3 results, using leave one subject out	201
TABLE 6.21	SVM Hypothesis 1.2 results, using "paced gait" as the test set	202
TABLE 6.22	SVM Hypothesis 1.3 results, using "paced gait" as the test set	202
TABLE 6.23	SVM Hypothesis 1.3 results, using "free gait" as the test set	202
TABLE 6.24	CART Hypothesis 2 results, using modified leave one out	203
TABLE 6.25	SVM Hypothesis 2 results, using modified leave one out	203
TABLE 6.26	Naïve Bayes Hypothesis 2 results, using modified leave one out	203
TABLE 6.27	Neural Net Hypothesis 2 results, using modified leave one out	203
TABLE 6.28	CART Hypothesis 2 results, using leave one subject out	204
TABLE 6.29	SVM Hypothesis 2 results, using leave one subject out	204
TABLE 6.30	Neural Net Hypothesis 2 results, using leave one subject out	204
TABLE 6.31	Neural Network Hypothesis 3 results, by subject, using cross-validation	205
TABLE 6.32	Informative features, as identified by CART	206
TABLE 6.33	Neural Net Hypothesis 1 results, using 10-fold cross-validation, and the top two features	209
TABLE 6.34	Neural Net Hypothesis 2 results, using 10-fold cross-validation, and the top two features	209
TABLE 6.35	Neural Net Hypothesis 3 results, by subject, using 10-fold cross-validation, and the top six features	210
TABLE 6.36	Contrived Groupings	211
TABLE 6.37	Neural Net results, using 10-fold cross-validation, and the top two features, with "Contrived Groups A"	211
TABLE 6.38	Neural Net results, using 10-fold cross-validation, and the top two features, with "Contrived Groups B"	212
TABLE 6.39	Neural Net results, using 10-fold cross-validation, and the top two features, with "Contrived Groups C"	212
TABLE 8.1	Comparison between the GaitShoe and the MGH Biomotion Lab	225
TABLE B.1	Information about volunteers for the subject testing	237
TABLE C.1	Sample results presented in a confusion matrix	240
TABLE C.5	CART Hypothesis 1 results, by class, using leave one subject out	242
TABLE C.2	CART Hypothesis 1 results, using modified leave one out	242
TABLE C.3	Naïve Bayes Hypothesis 1 results, using modified leave one out	242
TABLE C.4	Neural Net Hypothesis 1 results, using modified leave one out	242
TABLE C.7	CART Hypothesis 1 results, using "free gait" as the test set	243
TABLE C.8	CART Hypothesis 1 results, using "distracted gait" as the test set	243
TABLE C.6	CART Hypothesis 1 results, by subject, leave one subject out	243
TABLE C.10	SVM Hypothesis 1.1 results, using modified leave one out	244

TABLE C.11 SVM Hypothesis 1.2 results, using modified leave one out	244
TABLE C.12 SVM Hypothesis 1.3 results, using modified leave one out	244
TABLE C.9 CART Hypothesis 1 results, using "paced gait" as the test set	244
TABLE C.13 SVM Hypothesis 1.1 results, using leave one subject out	245
TABLE C.14 SVM Hypothesis 1.2 results, using leave one subject out	245
TABLE C.15 SVM Hypothesis 1.3 results, using leave one subject out	245
TABLE C.16 SVM Hypothesis 1.1 results, using "free gait" as the test set	245
TABLE C.19 SVM Hypothesis 1.1 results, using "distracted gait" as the test set	246
TABLE C.20 SVM Hypothesis 1.2 results, using "distracted gait" as the test set	246
TABLE C.21 SVM Hypothesis 1.3 results, using "distracted gait" as the test set	246
TABLE C.17 SVM Hypothesis 1.2 results, using "free gait" as the test set	246
TABLE C.18 SVM Hypothesis 1.3 results, using "free gait" as the test set	246
TABLE C.22 SVM Hypothesis 1.1 results, using "paced gait" as the test set	247
TABLE C.23 SVM Hypothesis 1.2 results, using "paced gait" as the test set	247
TABLE C.24 SVM Hypothesis 1.3 results, using "paced gait" as the test set	247
TABLE C.25 CART Hypothesis 2 results, using modified leave one out	248
TABLE C.26 SVM Hypothesis 2 results, using modified leave one out	248
TABLE C.27 Naïve Bayes Hypothesis 2 results, using modified leave one out	248
TABLE C.28 Neural Net Hypothesis 2 results, using modified leave one out	248
TABLE C.29 CART Hypothesis 2 results, using leave one subject out	249
TABLE C.30 CART Hypothesis 2 results, by subject, using leave one subject out	249
TABLE C.31 SVM Hypothesis 2 results, using leave one subject out	249
TABLE C.32 SVM Hypothesis 2 results, by subject, using leave one subject out	250
TABLE C.33 Neural Net Hypothesis 2 results, using leave one subject out	250
TABLE C.34 Neural Net Hypothesis 2 results, by subject, using leave one subject out	250
TABLE C.35 CART Hypothesis 2 results, using "free gait" as the test set	251
TABLE C.36 SVM Hypothesis 2 results, using "free gait" as the test set	251
TABLE C.37 CART Hypothesis 2 results, using "distracted gait" as the test set	251
TABLE C.38 SVM Hypothesis 2 results, using "distracted gait" as the test set	251
TABLE C.39 CART Hypothesis 2 results, using "paced gait" as the test set	251
TABLE C.41 Neural Network Hypothesis 3 results, by subject, using cross-validation	252
TABLE C.40 SVM Hypothesis 2 results, using "paced gait" as the test set	252
TABLE C.42 CART Hypothesis 3 results, by subject, using cross-validation	253
TABLE C.43 Naïve Bayes Hypothesis 3 results, by subject, using cross-validation	253
TABLE D.1 Insole sensor mapping	259
TABLE D.2 Vendor information	271

Chapter 1

INTRODUCTION

Clinical gait analysis is the investigation of the pattern of walking. At present, gait analysis is primarily carried out in one of two ways: in a motion laboratory, with full analysis of the motion of all body segments using highly accurate computer-based force sensors and optical tracking systems, or in an office with the clinician making visual observations. The first method is expensive, requires the maintenance of a dedicated motion lab, and uses cumbersome equipment attached to the patient, but produces well-quantified and accurate results for short distances. The second method is inexpensive and does not require any equipment, but the results are qualitative, unreliable, and difficult to compare across multiple visits.

There is a need for a low cost device that falls in between these two methods, and is capable of providing quantitative and repeatable results. In addition, there is a need for long term monitoring of gait, as well as quick diagnosis of chronic walking problems. Also, there is a need to be able to quantitatively analyze gait for patients who do not have access to motion analysis labs, such as is the case in economically disadvantaged locations.

This thesis discusses the development of an on-shoe system for continuous monitoring of gait. This system includes an instrumented insole and a removable instrumented shoe attachment. The data are sent wirelessly, providing information about the three-dimensional motion, position, and pressure distribution of the foot. The system was independently calibrated and analyzed, and was tested on fifteen subjects. The results from these

subjects were compared to the results from the gait analysis system at the Massachusetts General Hospital (MGH) Biomotion Lab.

1.1 Thesis Statement

The goal of this thesis was to design, build, calibrate, analyze, and use a wireless wearable system capable of measuring an unprecedented number of parameters relevant to gait. The system was designed to collect data unobtrusively, and in any walking environment, over long periods of time. It was built to be worn on the shoes, without interfering with gait. The sensors were calibrated, and the calibrated data were analyzed for information about the gait of the user, and the results of the gait analysis were validated against results from the optical tracking system in use at the MGH Biomotion Lab. The calibrated data were also used to generate features, which were used to classify the gait of the subject, using standard pattern recognition techniques. The system was also used to investigate real-time therapeutic feedback.

1.2 Motivation

Quantitative evaluation of gait is currently limited by the availability and the size of motion analysis labs. Motion analysis labs are expensive to maintain, and are typically only found in hospitals in large urban areas. Typically, patients can only walk about 7-10 meters per trial, and have one chance per trial to step on a disguised force plate. Alternatively, many physicians and physical therapists rely instead on observational gait analysis to evaluate patients. While well-trained medical specialists are undoubtedly capable of discerning a great deal of information about their patients' gait, small changes may be hard to detect, and a qualitative observation is difficult to compare between office visits or different specialists. Evaluation of common podiatric problems would be enhanced by an inexpensive method of quantitative evaluation. For instance, people with diabetes are often fitted with orthotics to improve their gait and reduce their chances of developing ulcerations on their feet; a straightforward and repeatable method of evaluating gait before

and after use of an orthotic would be desirable to optimize its shape and placement. Historically, orthotics have not been designed following systematic design procedures, but in an *ad hoc* manner, relying on the individual expertise of the orthotist [1].

In addition, gait and changes in gait are surrogate markers for a variety of other medically important phenomena: developmental maturation, likelihood of falling, and recovery from a stroke. Change in gait over extended time is used in neurological exams to diagnose dementias, and can be used to assess the adequacy of pharmacologic therapy in a number of neurologic/psychiatric disorders.

Finally, the development of a wearable wireless system has been greatly enabled by the many recent and on-going advances in sensor technology that have resulted in sensors which are small and inexpensive.

1.3 Project Description

The research sought to create a system that will provide instrumented gait analysis outside of traditional, expensive motion labs. Such a system has the potential to be highly informative by allowing data collection throughout the day in a variety of environments, thus providing a vast quantity of long-term data not obtainable with current gait analysis systems.

The top-level functional requirements for this system are:

1. Effect no change in gait.
2. Characterize the motion of both feet.
3. Be untethered.
4. Allow the subject to use his or her own shoes.

To meet these requirements, an on-shoe system has been designed and developed. The on-shoe components were configured in such a way that gait was minimally affected, and such that they could be readily fixed to a variety of typical walking shoes. The system was replete with sensors, with the goal of measuring more parameters than would otherwise be

necessary for any one application, essentially providing a wearable podiatric laboratory. A power source was contained on-shoe, and the system used wireless protocols to communicate between shoes and to transmit the data to a base-station; no cables of any sort were attached to either shoe.

This research evaluated the system both in persons with normal gait, and in elders with Parkinson's disease (PD). Subjects with PD were included for the purpose of evaluating the data in a population with altered gait. As indicated in recent research, the PD population would benefit from having a system which would allow evaluation of gait at home, by providing better information about gait abnormalities present in everyday life that have not traditionally been captured in analyses carried out in motion laboratories [2]. For example, this could provide the ability to titrate medication doses to the patient's current, rather than the average, needs.

1.4 Important Gait Parameters

As mentioned above, this thesis sought to create a system capable of providing clinically relevant information about gait. "Clinically relevant" is, of course, a subjective term which is certainly defined in many different ways. Therefore, to direct the design of the system, certain parameters of gait were identified (through a review of the literature about gait analysis and meetings with the physical therapists in the MGH Biomotion Lab) as important for the system to measure. The following gait measurements were identified:

1. Heel strike timing.
2. Toe off timing.
3. Dorsi-/plantar- flexion.
4. Stride length.
5. Stride velocity.

The system described within this thesis is designed so that, at a minimum, it is capable of characterizing these specific parameters of gait. The results for these and other parameters were compared to those obtained by the system in use at the MGH Biomotion Lab;

this comparison was used to validate the system and to make a statement about the clinical relevance of the information resulting from the on-shoe system.

1.5 Summary of Contributions

The work completed for this thesis has resulted in the following:

1. A robust wireless two-shoe system, capable of measuring many gait-relevant parameters, and including shoe attachments, insoles, and base-station.
2. Data collection from ten subjects with normal gait, and from five subjects with Parkinson's disease (PD).
3. Techniques for calibration of the system.
4. Methods for analysis of gait features, including "important gait parameters" (heel strike timing, toe off timing, dorsi-/plantar- flexion, stride length, and stride velocity).
5. Determination of features from the sensor outputs, classification of gait as gait of healthy subjects or gait of subjects with Parkinson's disease, and classification of the gait of ten individual subjects.
6. Identification of two features which distinguish normal gait from the gait of subjects with Parkinson's disease.
7. Investigation into the use of interactive real-time therapeutic feedback.

Chapter 2

BACKGROUND

To provide an understanding for the need of a wireless gait system, this chapter discusses the prior work in the field, as well as the current state of the art. In addition, the clinical need for this system is discussed.

2.1 Prior Work

There is extensive prior research investigating alternatives to the traditional motion lab for gait analysis. The obvious advantage of directly measuring the pressure distribution beneath the foot has driven many of the early shoe-based systems. The shrinking size of data storage has further encouraged the development of non-tethered systems.

2.1.1 On-Shoe Research Systems

Efforts to take measurements more directly at the foot interface go back to at least the 1960's, with most early work focusing on various pressure sensors on an insole to gauge the pressure distribution beneath the foot.

While there are obvious advantages in taking measurements directly, there are some potential disadvantages with instrumented shoes. For instance, if accurate measurements of pressure underneath anatomical landmarks are required, sensor placement must either be guessed at, or an initial test must be done to determine correct placement. The placement of the sensors must be durable enough to prevent movement within the shoe during

walking. The sensors themselves must be robust enough to withstand the normal and shear forces of walking, as well as the warm, humid climate inside the shoe. In addition, consideration must be made so that the instrumentation itself does not affect the gait. These limitations need to be taken into consideration during the design of any instrumented shoes [3].

Instrumented Insole for Pressure Distribution

In 1990, Wertsch *et al* [4] developed an exceptional system for measuring the pressure distribution beneath the foot. They first had each subject walk on inked paper to determine the locations of seven high pressure points corresponding to the five metatarsal heads, the big toe, and the heel center. They then placed seven force sensitive resistors (FSRs) at these locations, creating a specific insole for each foot of each subject. A seven channel amplification circuit was attached to each lower leg, and one shielded cable ran up each leg to a belt anchor. Both of the shielded cables were 10 m in length, and extended to connect to the analog-to-digital converter in a PC and to a power supply for the amplification circuit. Computer software was developed to collect and store the data, as well as to display the readings of all fourteen sensors in real-time, in two formats: bar graphs showing the pressure amplitude, and strip charts showing pressure vs. time. Although limited by requiring the subject to be tethered, this system gave detailed information about the pressure distribution beneath the foot, and provided those results in real-time.

Data collected with their device has led to a number of papers, including one quantifying the differences between shuffling and walking [5], and between sensate and insensate (no or little sensation in the foot) subjects [6]. In the latter study, the results led to a caution against drawing conclusions from a short segment of gait analysis in patients with sensory impairment, as a large step-to-step variation was found in these patients. This further emphasizes the need for a device capable of collecting data over a long time period.

Instrumented Insole for Gait Timing

In 1994, Hausdorff *et al* [7] developed a simple standalone "footswitch" system capable of detecting several of the temporal gait parameters. Their system consisted of two FSRs on an insole. The insole was cut from tracings of the subjects' feet on a manila folder, and the two FSRs (each square, 1.5 inches per side) were positioned under the heel and in the general area under the toes and metatarsals. The initial work used a circuit with a battery and data storage that was placed in the pants pocket of the subject; following work resulted in a single pack worn on the ankle [8] [9]. After collecting data, it was analyzed and compared to data taken simultaneously on commercial force plates. Calculations by their device found stance duration to be within 3% and swing and stride duration within 5% as compared to the results from the force plate.

Because the outputs of the FSRs were connected in parallel for hardware simplicity, they act as a single combined sensor. This does not affect the case where both sensors are active, or where both sensors are not active. However, this results in a loss of information if only one of the sensors is active, because it cannot distinguish between the two. For the calculations of gait timing, they did not find this to be a drawback; however the outputs of the FSRs on our insole were not combined, so that all the information can be utilized.

They have used the data from their insole to find patterns in gait [10], which they have been able to use to predict the maturation of gait in children [8], and the likelihood of falling in the elderly [9]. This simple device demonstrates that with only two FSRs, some types of abnormalities in gait can be distinguished from normal gait. This device is currently limited by the lack of real-time feedback.

Instrumented Insole for Conditions at the Foot Interface

More recent work resulting in shoe-based sensor systems with increasingly sophisticated measurement capabilities have been driven by sub-specialty interests in gait analysis. For diabetics, Morley *et al* [11] have developed an insole-based system to quantify the condi-

tions inside the shoe, with the goal of being able to predict progression of skin breakdown and ulceration in diabetic patients with peripheral neuropathy.

The laminated insole developed by Morley *et al* had pressure, temperature and humidity sensors designed to investigate the conditions at the foot interface. Combined pressure and temperature sensors were located beneath the heel and the region of the medial metatarsal head, pressure sensors were additionally located in the region of the central and lateral medial metatarsal heads, and a single humidity sensor was located centrally at the toes. Flexible wiring connected the insole to an electronics module and two AA batteries. These were located in a plastic enclosure, which was strapped to the calf of the subject. The data were stored on-board and uploaded to a computer via the serial port. It can currently store 4.5 hours worth of data, but with the implementation of data compression schemes, the data storage is expected to increase to 12-16 hours, to be able to cover a full day. They foresee a potential use of the device as an activity monitor for patients with diabetes, coronary heart disease, and/or obesity, to see if the subjects meet prescribed activity levels.

In initial work with their device [12], they were able to detect quantitatively distinct variations in pressure patterns that corresponded to different activities, and were able to correlate their results with previous studies. They have not yet published work investigating the tracking of the temperature and humidity sensors. Limitations of this device include restricted data storage capacity, a reported breakdown of connections, and the lack of real-time feedback.

Instrumented Insole and Shoe-based Gyroscope Device for Detection of Gait Timing

Another area of research driving devices capable of capturing information about gait is the development of neuroprosthetics used for walking assistance. Neuroprosthetics require inputs to trigger the functional electrical stimulation (FES) used to assist the patient in making the walking motions. Pappas *et al* [13] have developed a shoe and insole device capable of detecting four events during walking: stance, heel-off, swing, and heel-strike, as well as detecting whether the subject is walking or standing. Three FSRs are located on

an insole, one under the heel, and two at the inner and first and fourth metatarsal heads. The two FSRs at the metatarsal heads provided information about asymmetrical loading of the foot. The FSRs were taped onto a 3mm insole, and their positions were adjusted for each subject. Their system also included a gyroscope, which was attached to the back of the shoe, placed such that the sensing axis was perpendicular to the sagittal plane, providing measurements of rotation in the sagittal plane.

They implemented a pattern recognition algorithm with their system. In this algorithm, they divided the gait cycle into two distinct phases (stance, swing) and two distinct events (heel-off, heel-strike). There were seven possible transitions between these (stance to heel-off or directly to swing, heel off to swing or back to stance, swing to heel strike or directly back to stance, and heel strike to stance). Data from the FSRs and from the gyroscope were used to define the transitions. They verified their algorithm by comparing the data with results from a commercial motion analysis system using optical motion analysis (a Vicon 370 from Oxford Metrics Ltd.). In addition to testing their algorithm with walking and running speeds ranging from 0.5 to 12 km/hour, they challenged it with non-walking motions: sliding of the feet, standing up, sitting down, and shifting weight during standing. Their classification algorithm achieved a 99% detection rate for normal subjects and a 96% detection rate for subjects with impaired gait, as compared with the commercial system, with a detection delay of less than 90 ms. These results demonstrate that on-shoe systems with gyroscopes and FSRs are able to achieve comparable results to commercial optical systems.

More recent work [14] has resulted in an insole-only system where the gyroscope and a microcontroller have been embedded in the insole. Using the results from their previous work, the system was used on two subjects with incomplete spinal injury resulting in drop-foot. The system was used to trigger functional electrical stimulation (FES), and they were able to demonstrate a functional benefit of using it, for both subjects, while walking horizontally, uphill, downhill, and while sitting and standing.

This device was developed in order to detect two specific phases (stance, swing) and two specific events (heel strike, heel off) which occur during gait, in order to accurately trigger the electrical stimulation by neuroprosthetics used for walking assistance. As such, it only measures rotation of the foot in the sagittal plane. However, the system described within this thesis will be capable of quantifying motion of the foot in three axes, so it will include two additional gyroscopes, as well as three axes of accelerometers.

2.1.2 Off-Shoe Approaches

In addition to research into on-shoe devices, there has also been work in developing different types of instrumented laboratory spaces. One method of deriving more information about the foot is to instrument the surface on which the subject walks. This approach can be used both in the gait lab, as well as in the clinical setting. Cutlip *et al* [15] have developed an instrumented walkway 4.6 m in length, and have demonstrated the ability to calculate correct values for step period, stance duration and swing duration. Their system is also capable of calculating step length and stride velocity, although it was more accurate at low speeds, and less accurate when the subjects walked more quickly. Giacomozi and Macellari [16] have developed a "piezo-dynamometric platform" which can be used instead of a force platform. They have shown their system to be highly accurate at calculating the center of pressure of the foot. The ability to measure the pressure distribution at the floor interface allows users of this system to walk without any hardware attached to their shoes or feet. However, these types of systems constrain the walking distance of each trial to the length of the measuring platform. Also, these types of systems do not provide any information about the motion of the foot above the platform.

2.1.3 Gait Recognition Systems

A number of research platforms have been developed to recognize gait without instrumenting the subject. Analysis of videotaped subjects is of particular interest, and has received significant funding from DARPA for the "HumanID at a Distance" program, for potential use as a biometric identifier; earlier work on video analysis of gait was done at

the MIT Media Lab by Jim Davis [17]. At the University of Southampton, UK, researchers have been able to recognize subjects from videos, with results better than 80% for walking gait, and better than 90% for running gait, in a study with 20 subjects [18]. Researchers at the MIT AI Lab have achieved similar results, with better than 84% recognition using videos of walking gait, over 25 subjects [19]. At Georgia Tech, work is in progress both using video analysis [20], as well as a separate initiative using radar to analyze the gait cycle [21]. In addition the Aware Home project at Georgia Tech has resulted in a "Smart Floor" that includes ten tiles, each supported by four industrial load cells. The ground reaction force profiles measured across the tiles were capable of correctly identifying subjects 90% of the time, from a sample population on the order of ten people [22]. A group at the University of Oulu, Finland, also used a pressure-sensitive floor to recognize gait; with three successive footsteps, they were able to recognize gait correctly 89% of the time, for a sample of eleven people [23].

2.1.4 Commercial Systems

A variety of shoe interfaces have been developed commercially, with a wide range of applications. Taptronics developed a dance interface with a pair of piezoelectric tap detectors at the toe and the heel [24]. Force sensors have been used by ProBalance [25] for analysis of the golf swing, while inertial sensors have been used by Acceleron [26], Reebok (the Traxtar) [27], FitSense [28], and other companies for other athletic applications (primarily for runners). An example of a runner-specific shoe is the Raven from Vectrasense, a running shoe that detects whether the user is running or walking, and adjusts an air bladder within the frontal area of the shoe, such that the air bladder is filled for running to provide more support, and the air bladder is emptied slightly for walking to provide more cushioning [29]. A product poised to become available in late 2004 is the "1" from Adidas, a running shoe with an on-board microcontroller, Hall effect sensors to measure the change in heel compression, and a motor to adjust tension in a stainless steel cord in the heel to achieve the user's desired heel compression [30] [31].

For medical applications, Tekscan and Clevemed, among others, have developed insoles which measure pressure distribution [32] [33]. NCSA's Cyberboots use a pressure sensor array in an overshoe to provide walking interaction in a virtual reality environment [34]. In addition, MiniSun markets "The IDEEA LifeGait System", which uses the outputs of accelerometers placed on various parts of the body with "artificial intelligence" algorithms to determine a number of parameters relating to gait and motion [35].

Of all these products, FitSense and Acceleron have developed systems most closely related to this research [26] [36] [37]. Acceleron has developed a sensor which attaches to the laces or within the insole of a shoe, and measures linear acceleration in three axes, transferring the data wirelessly and in real-time; they have obtained two patents on their technology. In the patents, they describe the use of accelerometers and rotation sensors in conjunction with an electronic circuit which carries out math calculations. They detail the methodology and the equations they use to calculate the distance, speed, and height jumped. In the later patent [37], they describe using radio frequency to send data from the sensors to a wristwatch or a remote device; this patent also describes the inclusion of a GPS device for direction and location information; details about the accuracy of the measurements and calculations were not available. The FitSense FS-1 system similarly attaches to the laces of a shoe, and transmits distance and speed to a watch (it also has an optional heart rate monitor); it has a reported accuracy of 98% [28]. While these systems accomplish some of the goals of this research project, neither system has the extent of sensors used in this research, and neither attempts to fully describe the gait in a manner that can be used as a clinical supplement to the motion analysis laboratory, and could be developed for a recreation sports product.

2.1.5 Expressive Footware: Instrumented Insole and Multiple Shoe-Based Sensors

The work in this thesis developed from the Expressive Footware project developed by Dr. Joseph Paradiso, and students in the Responsive Environments Group at the MIT Media Lab [38]. The Expressive Footware project resulted in a pair of running shoes that were

each equipped with a wireless sensor board and an instrumented insole. Each insole measured dynamic pressure at the heel, bidirectional bend of the insole, the height of each foot above a conducting mat on the floor, and had three FSRs: two placed roughly under the medial and lateral metatarsal heads (to allow the dancer easy control by leaning left or right), and one outside the shoe, mounted at the toe. Each sensor board was permanently attached to the lateral side of the shoe, and contained a gyroscope for the angular rate of the foot about the vertical axis, a three-axis compass to determine the orientation of the foot relative to the Earth's local magnetic field, two axes of acceleration (the two axes in the plane of the sensor card), and three axes of shock acceleration. Finally, an integrated sonar receiver on each sensor board, in conjunction with four sonar transmitters on the floor, provided the position of each foot in the plane of the floor. This system was built for control, not for measurement; the sensor outputs were not saved for analysis, but were used to directly control real-time musical outputs, generated by a computer that interpreted the basestation data stream with an elaborate rule base.

This highly instrumented shoe was worn by dancers and the outputs of the sensors were used to interactively control music. It was completely wireless, with all hardware located directly on the shoe, and provided real-time control of the musical mappings. It reached high acclaim in the dance community, and was recognized with the *Discover Award for Technical Innovation* in 2000. It is shown in Figure 2.1, with the shoe hardware in the foreground, and sensor outputs in the background.

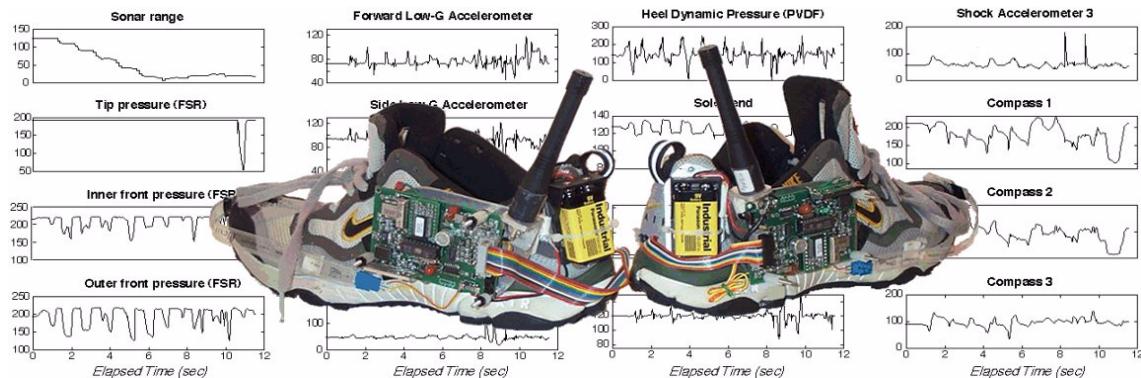


Figure 2.1 Expressive Footware and its sensor outputs

After using this device extensively in interactive dance, Prof. Paradiso became interested in further developing it as a medical tool for quantifying the motion of the foot. This interest, combined with interest from collaborators at the MGH Biomotion Lab led to the initiation of this thesis.

2.2 Current State of the Art

Clinical gait analysis is currently carried out in two very different ways. One is visual observation, and the other is analysis in a motion laboratory.

2.2.1 Observational Gait Analysis

Observational gait analysis (OGA) consists of a well-trained physician or physical therapist assessing patients by watching them walk, either in real-time or on a videotape. This method requires no specialized equipment other than a video camera and no cost beyond the clinician's time and training; however, it is entirely qualitative.

A study on the reliability of OGA in children with lower-limb disabilities examined the ratings on a three point scale, as rated by three experts observing fifteen subjects on videotape; the raters agreed on fewer than 7 ratings out of 10 [39]. Another study looked at the reliability of analyzing knee motion of three different subjects with gait changes due to rheumatoid arthritis, on videotape, as rated by fifty-four licensed physical therapists; this study found only slight to moderate agreement between the raters [40].

A recent effort reviewed fourteen studies (including the two mentioned above) that investigated the reliability of OGA, and found that the majority of studies concluded that qualitative observation of gait has poor to moderate reliability [41]. The authors also evaluated OGA, using eighteen physical therapists who routinely use OGA to assess changes in gait following a stroke. The therapists were shown a video, and asked to evaluate ankle power generation by rating each subject on a 22-point scale; the video included an audible tone at heel strike. The subjects were also evaluated using reflective markers and a camera-based

motion analysis system. Under these highly systematized evaluation conditions, the therapists were able to demonstrate moderate to high reliability in their ratings, as compared with the results from the motion analysis system.

2.2.2 Overview of Clinical Gait Analysis

Comprehensive gait analysis is generally used for the assessment of a patient with a movement disorder. There are as many as five components to the gait analysis [42]:

1. *Videotape Examination*: to observe gait abnormalities [in slow motion or freeze-frame]
2. *Temporo-Distance Parameters*: cadence, stride length, speed [may be measured manually]
3. *Kinematic Analysis*: measurement of movement [usually measured with cameras, LEDs, IR]
4. *Kinetic Measurement*: forces between foot and ground [usually measured with force plate]
5. *Electromyography*: electrical activity of muscles [surface or fine wire electrodes]

All components are not necessarily used, especially when a motion analysis lab is utilized, as often the bulk of the gait analysis is performed employing the data from the kinematic analysis and kinetic measurements. Data from these can also be used to calculate the temporo-distance gait parameters. Electromyography (EMG) is used less often than the other techniques; surface electrodes may have trouble sensing deep muscles and are less accurate than the data obtained from fine wire electrodes, which can be painful to the patient [43].

While this is a very accurate method of measuring all the parameters of gait, it requires expensive equipment, and a dedicated lab space, usually a minimum of 10 meters x 10 meters. This size means that subjects cannot walk very far before stopping and turning around. Investigators have found that, in general, a minimum of two trials are needed and better results are achieved when data from multiple trials are averaged, since data from a single trial are too variable to rely on alone [44]. In addition, the subject must step directly

on the force plate in order to obtain an accurate measurement; however, asking the subject to aim for the force plate may result in an alteration of the subject's gait, called "targeting."

The following sections discuss different methods used for kinematic analysis; the technology has been driven in part by the computer graphics and animation industries. For example, a group at Laboratoire d'Electronique de Technologie de l'Information (LETI), France, is investigating the use of three axis sensors (with accelerometers and magnetometers) for the purpose of using these small sensors in wearable clothing to improving motion capture for 3D virtual worlds [45].

2.2.3 Optoelectronic Systems

An optoelectronic system involves placing light emitting diodes (LEDs) on the subject. The LEDs are turned on sequentially by a computer, and viewed by a camera. Because the computer triggers the LEDs, there is no question about which LED is viewed by the camera at a given time point. However, reflection off the floor, or other surfaces reflective to infrared, such as human skin, can reduce the accuracy of the system.

The system in use at the Massachusetts General Hospital (MGH) Biomotion Lab uses a Selspot II system (Selective Electronics, Partille, Sweden) to serially sample up to 64 infrared LEDs, arranged in arrays, at a rate of 153 Hz. The LED arrays are placed on eleven body segments (bilaterally: feet, shanks, thighs, arms; and, the pelvis, trunk, and head). The TRACK kinematic data analysis software package is used to generate photo-stereogrammetric reconstruction of the 3-D positions of the LEDs and to define the six degree of freedom kinematics of the arrays [46]. Within the viewing volume¹, this system is capable of accurately defining the 3D positions of each body segment to within 1 mm, and the three orientations to within 1 degree, though actual results during testing may vary. With the technology currently in use, this system requires the subject to be wired

1. The viewing volume is the area of the room visible to the cameras that sample the LED output; in the Massachusetts General Hospital Biomotion Lab, the viewing volume has a width just under 2 m along the direction of forward gait.

("tethered") to the computer, though it is likely possible to convert the system to a wireless system.

In addition, two Kistler piezoelectric force plates (Kistler Instruments Type 9281A, Winterthur, Switzerland) are used to acquire ground reaction forces; this system has an accuracy of $\pm 1\%$ of full scale; as set in the MGH Biomotion Lab (BML), this corresponds to ± 10 N of vertical force, and ± 5 N of shear force, for forces and frequencies encountered during gait (the unloaded force plate is recalibrated to a load of 0 N after each gait trial) [47] [48]. A photo of a subject instrumented with the MGH Biomotion Lab equipment is shown in Figure 2.2.



Figure 2.2 A subject at the MGH Biomotion Lab

2.2.4 Videographic Systems

Systems using videography with reflective markers are the most frequently used system in motion analysis labs. This type of system involves placing markers which are highly

reflective on the subject. The markers are illuminated and viewed by cameras; illumination is generally achieved with infrared lights or an infrared strobe located near the camera. This type of system allows the subject to walk untethered, however, not all markers may be illuminated at a given time point. This results in the need for significant post processing to sort and identify the markers.

Two major manufacturers of such systems are Vicon Motion Systems and Motion Analysis Corporation. Vicon systems can be set up with as many as 24 cameras; the top of the line M2 camera has 1280 x 1024 resolution (with a digital CMOS sensor), and can capture up to 1000 frames per second [49]. Motion Analysis Corporation has more than 600 systems installed in motion laboratories worldwide. Its premier system, the Eagle Digital, can be set up with as many as 64 cameras, and also has a 1280 x 1024 CMOS sensor. At this resolution it can capture 480 frames per second [50].

2.2.5 Electromagnetic Systems

Electromagnetic systems involve having a stationary transmitter which emits a magnetic field, and instrumenting the subject with electromagnetic coils, which detect this field. A benefit of this type of system is that there are no "line of sight" requirements, as the relatively low-frequency magnetic field lines easily penetrate human tissue and non-conductive objects. However, the receivers must be within the range of the transmitter. At this point, electromagnetic systems are not widely used in gait analysis, most likely because the systems currently available only track a small number of points. In addition, the electromagnetic field is vulnerable to distortion by magnetically susceptible materials in the vicinity of the system. However, these systems are currently used in other areas of motion research, such as hand or head tracking, and may be of interest for gait analysis when they have the ability to track a greater number of points.

The two best known systems of this type are made by Polhemus and Ascension Technology Corporation. Polhemus uses an AC magnetic field; its FASTRAK® system is advertised as having an accuracy of 0.03 inches RMS for position and 0.15 degrees RMS for

orientation, and a resolution of 0.0002 inches and 0.025 degrees per inch distance from the transmitter (4-6 feet are recommended, but up to 10 feet is possible). It can track up to four sensors per transmitter, and up to four transmitters can be used at once, providing the ability to track sixteen sensors [51] [52]. Ascension Technology Corporation uses a pulsed DC magnetic field; its Flock of Birds® system can track up to four sensors, and is advertised as having an accuracy of 0.07 inches RMS for position, and 0.5° RMS for orientation, with a resolution of 0.02 inches and 0.1° when 12 feet from the transmitter. Alternatively, Asencion's MotionStar Wireless® system can track up to twenty sensors, and is advertised as having an accuracy of 0.6 inches RMS for position, and 1.0° RMS for orientation, with a resolution of 0.1 inches and 0.2° when 10 feet from the transmitter. [53].

2.3 Clinical Need

The current clinical methods of analyzing gait fall at two extremes - on one end is observational gait analysis, which is inexpensive but qualitative. At the other end is analysis in a motion laboratory, which is quantitative but expensive. In both methods, the subject is very aware of being observed and analyzed, which is likely to affect the gait of the subject.

An on-shoe system could provide the benefits of both methods without the drawbacks. It could be far less expensive than the motion lab, and could provide quantitative output about the gait. If unobtrusive, it could measure the gait of the subject while the subject is unaware of being tested, over an extended period of time.

In addition, an on-shoe system could ultimately provide unique features. Since it is mounted on the shoe of the subject, the patient can be sent home to monitor gait throughout the day, in a variety of environments, and in a variety of situations. In addition, it could analyze the gait in real-time, which would allow it to provide the wearer feedback of various types (e.g. musical, tonal, visual, tactile, electro-stimulation), which could be useful for physical therapy or gait training. There are many types of patients who may benefit from such real-time feedback; the subject testing in this thesis included investigation of the gait of subjects with Parkinson's disease (PD). Finally, an on-shoe system could be

used in localities (e.g. rural areas or third-world countries) where patients do not have access to a motion lab facility.

2.3.1 Utility of On-Shoe Device in Subjects with Parkinson's Disease

This instrumented shoe is likely to be useful for a wide range of gait conditions; in order to begin evaluation of subjects with altered gait, the subject testing for this thesis included five patients with Parkinson's disease (see Appendix A.2 for more information on Parkinson's disease). Future interests for this system include the ability to use this system to analyze the gait of PD subjects in their home environment, as well as to provide real-time auditory feedback to subjects with PD.

Changes in Gait Due to Parkinson's Disease

Hausdorff *et al* [54] investigated changes in gait variables in subjects with PD. Using their system described in Section 2.1.1 (see page 31), they recorded data from control subjects, PD subjects, as well as subjects with Huntington's disease. They found a statistical difference in the speed (1.35 m/sec for controls, 1.00 m/sec for PD) and in the "double support time," which is the duration both feet supported the patient simultaneously (305 msec for controls, 376 msec for PD). These are both parameters that can be easily measured by our device; these are important to keep in consideration when designing the study with PD subjects.

Changes in gait parameters between walking and shuffling were examined by Wertsch *et al* [5] using their system described in Section 2.1.1 (see page 30). Predictably, they found that the average velocity and stride length to be lower during shuffling (0.51 m/sec and 0.63 m) than during walking (1.29 m/sec and 1.55 m). They found that peak pressures were lower at all fourteen sensor sites (seven per foot) during shuffling, while foot-to-floor contact duration was increased at all fourteen sensor sites. The decreases in peak pressures ranged from 7.0% at the fifth metatarsal to 63.2% at the great toe, with a decrease of 41.6% in peak pressures summed over the entire foot. The increase in contact

duration ranged from 22% at the fifth metatarsal to 76.9% at the heel. The findings in their study may be of use in analyzing the data which will be collected from PD subjects.

A Clinical Need to Evaluate PD Subjects Outside the Motion Lab

Morris *et al* [2] have evaluated the biomechanics and motor control of gait in subjects with PD. Their paper includes a tabular listing of eighteen studies involving subjects with PD, and lists the medication state of the patients. Morris *et al* recommend a change from the current trend of lab based studies investigating PD subjects in straight line walking, towards studies in their homes and communities with more complex gait activity.

Effect of Rhythmic Cues on Subjects with Parkinson's Disease

A recent study found that auditory rhythm and the resulting physical response shows great potential as a therapeutic method for rehabilitating patients who have movement disorders [55]. The authors initially investigated the effect of "rhythmic auditory stimulation" (RAS), provided to thirty-one subjects with PD for three weeks. With RAS at 10% faster than each subject's baseline cadence, significant improvement was found in mean gait velocity, cadence, and stride length, both for the twenty-one subjects on medication, and the ten subjects off medication; in subjects on medication, the mean gait velocity increased by 36%, and in subjects off medication, the mean gait velocity increased by 25% [56]. The authors followed this with a twelve week study of twenty-one subjects; the subjects were pretested, used RAS, using audio tapes, for 30 minutes daily for three weeks, and were post-tested at seven weekly follow-ups, and a final post-test during the twelfth week (RAS was not used after the first three weeks). The gains in stride length, cadence, and velocity were maintained for 3-4 weeks after the training period, but most values returned to pretest values by the fifth follow-up week [57].

Another group studied the effects of "musical therapy" (MT), as compared to physical therapy (PT). Thirty-two subjects with PD were randomly split into two groups of sixteen: one group received weekly sessions of MT for three months, the other received weekly sessions of PT for three months. The MT sessions involved a variety of active music par-

ticipation, including choral singing, voice exercises, and rhythmic movement, and the PT sessions involved a variety of motions designed to improve balance and gait, including passive stretching and specific motor tasks.

The study used standard scales for rating the subjects: the Unified Parkinson's Disease Rating Scale: Motor Subscale (UPDRS-MS) was used with the Wilcoxon signed-rank test to evaluate the significance of changes in movement. The Unified Parkinson's Disease Rating Scale: Activities of Daily Living (UPDRS-ADL) and the Parkinson's Disease Quality of Life (PDQL) were used with the Mann-Whitney U test to evaluate the significance of changes in quality of life. The subjects in the MT group demonstrated significant improvement ($p<0.0001$) in UPDRS-MS scores, with particular improvement ($p<0.0001$) of bradykinesia (slowness of movement). Subjects in the PT group did not show significant improvement in either the UPDRS-MS scores or in bradykinesia. In addition, only subjects in the MT group showed significant improvement in UPDRS-ADL ($p<0.0001$) and in PDQL ($p<0.0001$). However, only subjects in the PT group had significant ($p<0.0001$) improvement in rigidity. In addition, at follow-up two months after the end of the study, the improvement in rigidity found in the PT group had persisted, while the parameters improved by MT had returned to their baseline values [58].

These initial studies investigating the use of auditory feedback for persons with PD suggest that such techniques may provide a method to improve motion, and even quality of life. The system described in this thesis could open up new avenues of therapy by allowing auditory feedback to be tailored in real-time to the movement of the subject, and by providing the subjects access to therapy in their home environments.

2.3.2 Studying Gait Outside of the Motion Lab

As mentioned in Section 2.3.1 (see page 45), subjects with PD would greatly benefit from a system capable of quantitatively analyzing gait outside of the motion lab. In addition, such a system could be very useful in areas where a motion lab is not accessible.

From the limited research available regarding gait analysis in third world countries or rural areas, it is clear that an inexpensive and quantitative method of studying gait is needed. For a study investigating recovery of gait after a stroke in Soweto, South Africa, a paper survey was used to ask patients about their recovery. In this survey, the ability to catch a taxi in Soweto was used as a measure to assess the patient's gait handicap [59].

A study in Germany investigated changes in gait in healthy subjects and in subjects with PD, with forty-three subjects recruited from an urban area (Berlin), and forty-seven subjects recruited from a rural / semi-urban area (Innsbruck and surrounding Tyrol). To evaluate gait, a system was used which involved attaching threads with Velcro straps to each foot at the second metatarsal head [60]. The threads were attached to a pulley system, and rotations of this system were measured by an optical recording device as the subject walked (the thread length allowed the subject to walk up to 10 m). All subjects were evaluated in a quiet environment with a gray colored walkway. This system provided measurements of parameters such as stride length, stride duration, and cadence. The study found that within each group (urban or rural), subjects with PD walked with a slower cadence than their healthy counterparts; additionally, subjects in the urban group (including those with PD) walked with a faster cadence than the subjects in the rural group [61].

A system which is not confined to evaluating gait within the motion lab would have great benefit in allowing better testing and evaluation of subjects who do not live near motion lab facilities. Judging from the lack of papers investigating the gait of subjects in rural areas, as well as the low levels of sophistication of the methods employed, such a system would open up a new venue of research. and greatly benefit patients who have no access to fully-equipped motion labs.

Chapter 3

HARDWARE DESIGN

The focus of this thesis was the design and implementation of the on-shoe system used for the measurement of gait. Section 3.1 discusses sensor selection, and Section 3.2 describes the physical implementation. Section 3.3 describes the function of each sensor in further detail. Section 3.4 describes additional electronics used in the system, and Section 3.5 summarizes the overall design of the "GaitShoe" system.

3.1 Sensor Selection

The first step in designing the GaitShoe was to select the appropriate sensors, with the goal of creating a highly instrumented system capable of sensing many parameters which characterize gait. As discussed in Section 1.4, several important parameters of gait were identified: heel strike timing, toe off timing, dorsi-/plantar- flexion, stride length, and stride velocity. Additional parameters of interest include global rotations of the foot, distance moved and velocity in the vertical and side-to-side axes, the pressure distribution underneath the foot, and orientation of the feet relative to each other.

Timing Parameters and Pressure Distribution

To assess the timing parameters and pressure distribution, force sensitive resistors (FSRs) and polyvinylidene fluoride (PVDF) strips were selected to be placed underneath the foot.

A FSR is a sensor whose electrical resistance decreases as the applied load increases. Two FSRs were placed underneath both the first and fifth metatarsal heads, and two more were placed medially and laterally underneath the heel pad. While use of only four force sensitive resistors will not provide a full picture of the force distribution beneath the foot, the number is sufficient to provide a general picture of medial vs. lateral force, and heel vs. metatarsal force. The assessment of heel vs. metatarsal force provides information to be used in determining stance time, and thus heel-strike and toe-off timing.

The PVDF strips are piezoelectric sensors, which were configured to provide an output corresponding to dynamic pressure. They were chosen for their fast response time, and were selected to be located directly beneath the heel and the great toe in order to provide additional information about heel-strike and toe-off timing.

Other force or pressure sensors were considered. In particular, fine-grain printed arrays of FSRs, such as Tekscan's F-Scan® system provide extensive information about the pressure distribution underneath the foot [32]; however, the high cost of Tekscan's proprietary system rendered it a prohibitive choice.

Flexion

Two bi-directional bend sensors were selected for use in analyzing flexion during gait. The resistance of the bend sensors changes as the sensor is bent. One of the bend sensors was located at the back of the heel, and held next to the shin by an ankle bracelet to provide information about plantar flexion and dorsiflexion. The second bend sensor was located in the insole approximately centered on the metatarsals, to provide information about flexion at the metatarsals.

Other methods of measuring dorsi-/plantar- flexion were considered, such as capacitance between foot and shin, or ultrasound measurements between the foot and the shin. Ultimately, a back-to-back pair of resistive bend sensors were selected due to their unobtrusiveness and ease of implementation. Fiber optic bend sensors such as those made by

Measurand® were also considered because of their high accuracy and resolution, but were prohibitively expensive [62].

Distances, Velocities, and Orientations

Three gyroscopes and two dual-axis linear accelerometers were chosen to be placed at the back of the shoe. By orienting the gyroscopes and accelerometers such that the individual sensing axes are aligned along three perpendicular axes, the angular velocity and linear acceleration can be measured in three-dimensions. Hence, velocity and stride length can be obtained respectively from single- and double-integration, with respect to time, of the acceleration component corresponding to forward motion of the foot. Displacements and velocities in the other two axes can be similarly acquired. The gyroscopes provide information about the rotation of the foot, which can similarly be integrated once with respect to time to provide the angle. The use of all six measurements can therefore be used to analyze the orientation of the foot; a device capable of all six measurements is called an "inertial measurement unit" (IMU).

Late in the project, two additional types of sensors were implemented: an electric field sensor and an ultrasound sensor. The electric field sensor was added to investigate the utility of using a more direct method of measuring the height of the foot above the floor, via capacitive loading [63]. Using multiple electric field sensors would allow the height of the foot to be measured at discrete locations, such as at the heel and the toes. The ultrasound sensor was added to provide a method of measuring the distance and relative orientation between the two feet. In addition, an ultrasound sensor could also be used to measure the height of the foot above the floor.

Summary

Table 3.1 summarizes the parameters of interest, along with the type of sensor(s) selected for each parameter, and the corresponding sensor output.

TABLE 3.1 Sensor selection

Parameter	Sensor	Sensor Output
Heel-strike timing and toe-off timing	FSRs, and PVDFs	<i>FSRs</i> : Resistance change corresponding to applied force across the sensor, resulting from change in displacement of the sensor. <i>PVDFs</i> : Voltage change corresponding to dynamic pressure across the sensor.
Dorsi-/plantar-flexion	Bend sensors	Resistance change corresponding to flexion angle, resulting from strain of the sensor.
Stride length and stride velocity	Accelerometers	Voltage change corresponding to acceleration; single integration of forward acceleration yields velocity, double integration yields distance (integration occurs after correcting for gravitational component).
Orientation	Gyroscopes	Voltage change corresponding to angular velocity; single integration yields angle of rotation.
Stance width	Ultrasound	Time of flight corresponding to distance.
Displacement side-to-side	Accelerometers	Voltage change corresponding to acceleration; double integration of lateral acceleration yields distance.
Height of foot above floor	Accelerometers, electric field, and ultrasound	<i>Accelerometer</i> : Voltage change corresponding to acceleration; double integration of vertical acceleration yields distance. <i>Electric field</i> : Capacitance corresponding to distance. <i>Ultrasound</i> : Time of flight of reflections, corresponding to distance.

3.2 Physical Implementation

In designing the GaitShoe hardware, the top-level functional requirements, as listed in Section 1.3, were considered:

1. Effect no change in gait.
2. Characterize the motion of both feet.
3. Be untethered.
4. Allow the subject to use his or her own shoes.

The most important requirement is the first, so in order not to cause changes in the gait, the hardware had to be small, compact and lightweight. Minimizing the weight is important because it has been shown that subtle effects on the gait occur when the lower-extrem-

ties are loaded with weights which are on the order of 1-2% (or greater) of the body weight [64]. Therefore, by keeping the mass of the final prototype under 300 g, it is not expected that the adult subjects would experience any change in gait due to the weight of this system; 300 g is 1% of 30 kg (66 pounds); subject testing was carried out on adults, all of whom weighed more than 45 kg (100 pounds).

Both shoes were instrumented, which satisfied FR2. To satisfy FR3, each shoe had its own power supply, and a wireless transceiver¹, based on radio frequency (RF), was used to transmit the data to a basestation connected to a laptop.

To meet FR4, the hardware was designed to be readily attachable to shoes and removable without causing any damage; and to comply with FR1, the attachment to the shoes was designed to not interfere with walking.

The design of the hardware needed to accommodate the sensors that must be located beneath the foot, all of the electronics (including additional sensors), an antenna for the wireless transmission, and the power supply. These requirements resulted in the design of the GaitShoe system, as shown in Figure 3.1.

The GaitShoe system was comprised of two shoe modules and a basestation. Each shoe module consisted of an instrumented insole placed beneath the foot, and an attachment which mounted to the back of the shoe. The instrumented insoles contained the force sensitive resistors, the polyvinylidene fluoride strips, one bend sensor, and part of the electric field sensor; the other bend sensor was connected to the insole, but placed behind the shin and held in place with an ankle strap, and an additional part of the electric field sensor was placed underneath the shoe. The shoe attachments contained a "stack" of printed circuit boards containing the IMU sensors, general electronics, the antenna, and the power supply, as well as the electronics and the hardware for the ultrasound sensor [65]. The

1. Another option for wireless data collection was to store the data on-board and download it later; however, wireless transmission was selected since it allowed the data to be analyzed in real time.

basestation received the data from both shoes, and transmitted the data to a computer via the serial port. Each part of the GaitShoe system is described below.

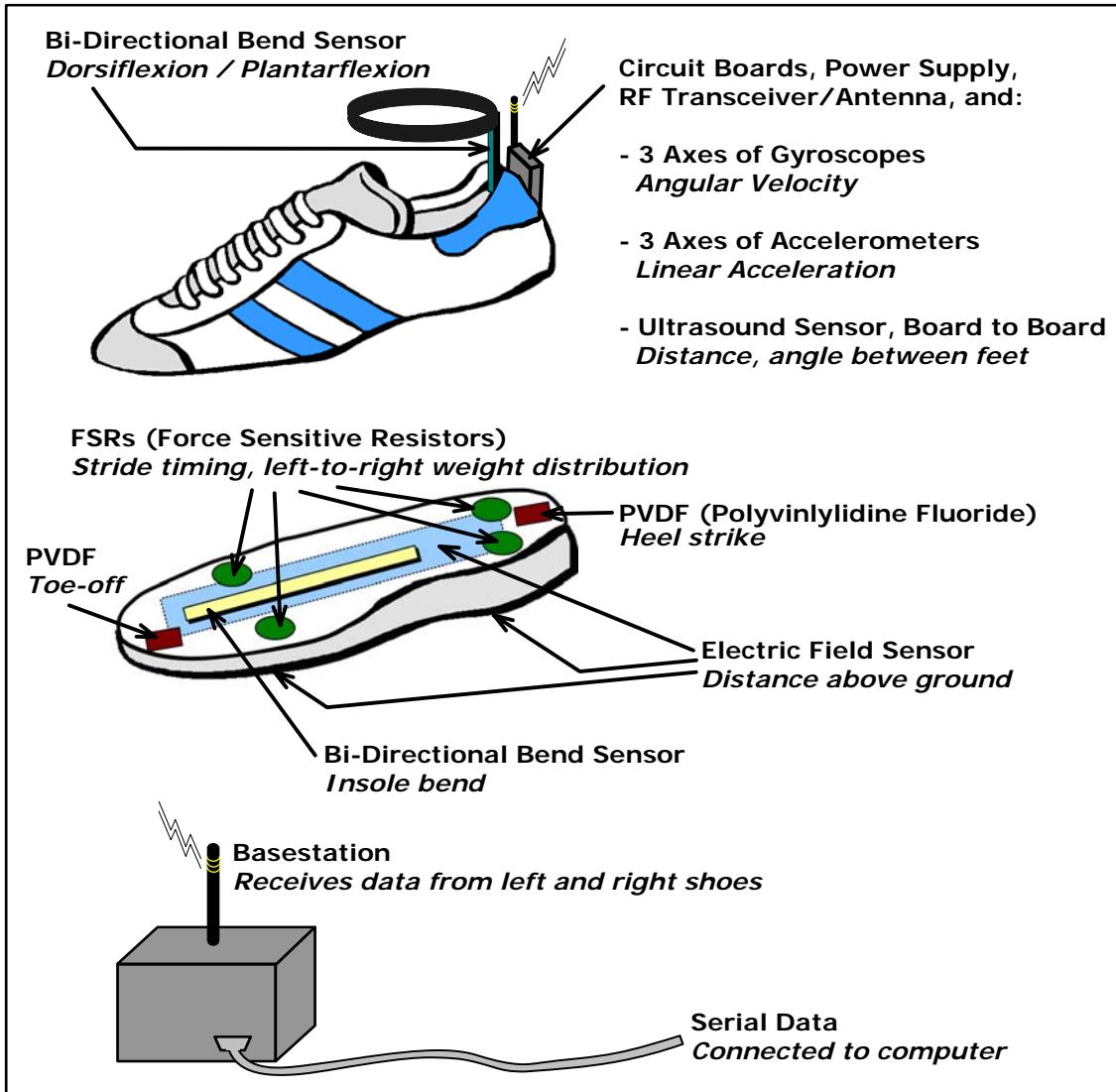


Figure 3.1 Schematic of the GaitShoe system

3.2.1 The Stack

The electronics used a stacking platform initially developed by Ari Benbasat [66], and redesigned to meet the requirements of the GaitShoe [65]. The idea was to make several small printed circuit boards which can be stacked together, resulting in a compact volume, rather than one large printed circuit board. The GaitShoe stack is shown in Figure 3.2.

Each individual circuit board contains sensors and electronics that meet a specific function; each board will be briefly discussed below, and the sensors and electronics are discussed in detail in the following sections.

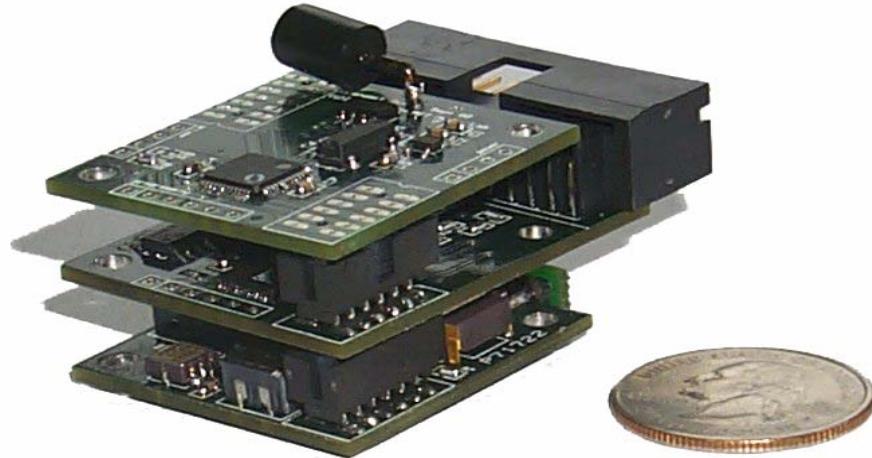


Figure 3.2 The stack

The Main Board

The Main board contained the microcontroller, wireless transceiver, antenna connection, and the connection for the inputs from the power board. It is shown in Figure 3.3 (attached to the Power board). The Main board controlled the collection and transmission of the data from all of the sensors.

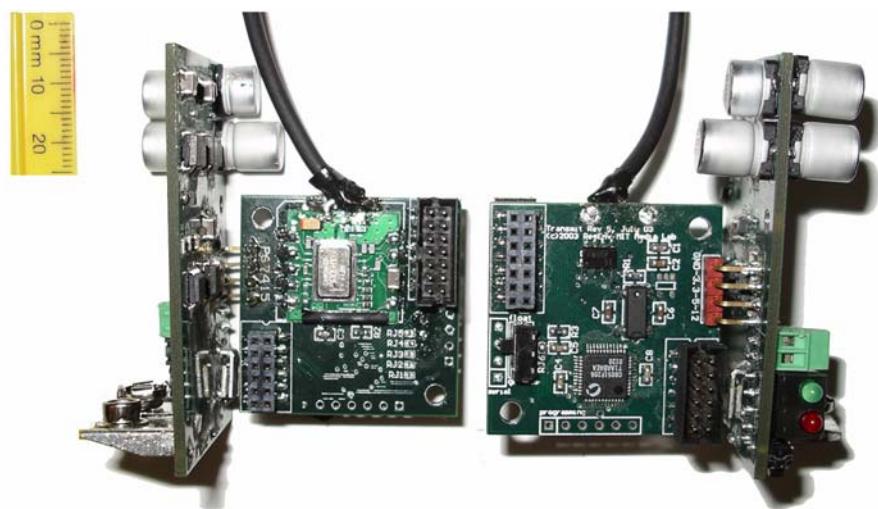


Figure 3.3 Photo of the front and back of the Main board

The IMU Board

The inertial measurement unit (IMU) board contained the two dual-axis accelerometers, and the three gyroscopes. The accelerometers and gyroscopes were oriented such that the board was capable of measuring angular velocity and linear acceleration about three axes. It is shown in Figure 3.4.

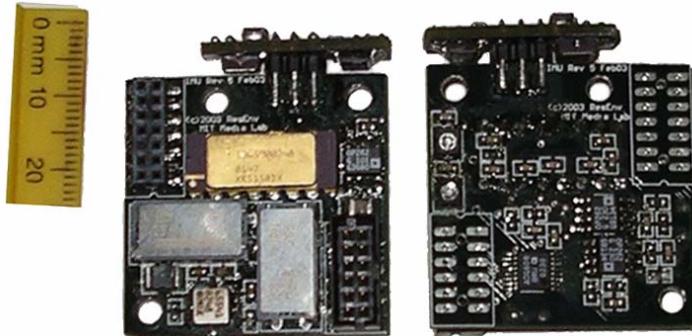


Figure 3.4 Photo of the front and back of the IMU board

The Tactile Board

The Tactile board is shown in Figure 3.5. It contained all the electronics for the force sensitive resistors, the bi-directional bend sensors, the polyvinylidene fluoride strips, and the electric field sensors. The electric field sensors connected to a header via co-axial cable, and plugged into the small receptacle at the side of the board. The rest of the sensors were connected to a header via ribbon cable, and plugged into the large receptacle at the top of the board.



Figure 3.5 Photo of the front and back of the Tactile board

The Ultrasound Boards

The Ultrasound boards, designed by undergraduate Steven Dan Lovell, contained the electronics for the ultrasound sensor. As shown in Figure 3.6, there were two versions, one each for the left and the right shoes. The ultrasound sensor measured the distance between and the angle between the two feet. The left ultrasound board contained an ultrasound transmitter and the corresponding electronics, and the right ultrasound board contained the connections for two ultrasound receivers and the corresponding electronics.

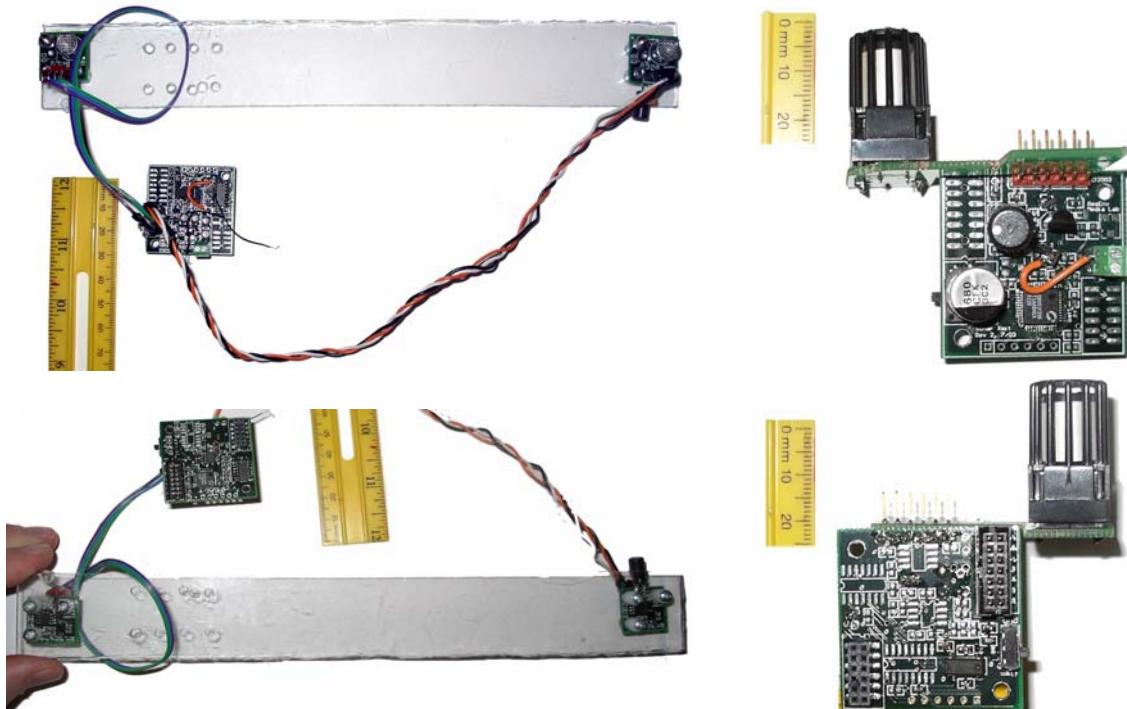


Figure 3.6 Photos of the front and back of the transmit Ultrasound board for the right foot (right photos), and the front and back of the receive Ultrasound board for the left foot (left photos)

The Power Board

The power board was developed to provide each GaitShoe system with an on-board power supply. It contained connections for a +9 V battery, an on-off switch, indicator lights, a fuse, voltage regulators for +3.3 V, +5 V, and +12 V, and a connector to transmit the ground and power lines to the main board. It is shown in Figure 3.7.

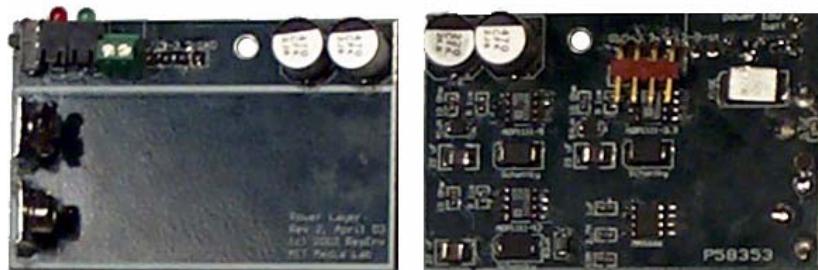


Figure 3.7 Photo of the front and back of the Power board

3.2.2 Insole Design

An insole was designed to accommodate the several sensors which were connected to the electronics via ribbon cable, as described in Section 3.2.1: The Tactile Board. The insole, shown in Figure 3.8 contained the four FSRs (yellow), the two PVDF strips (green), and one of the bi-directional bend sensors (magenta). The insole provided a straightforward method to position the sensors in the correct locations beneath the foot (see also Figure 3.1).

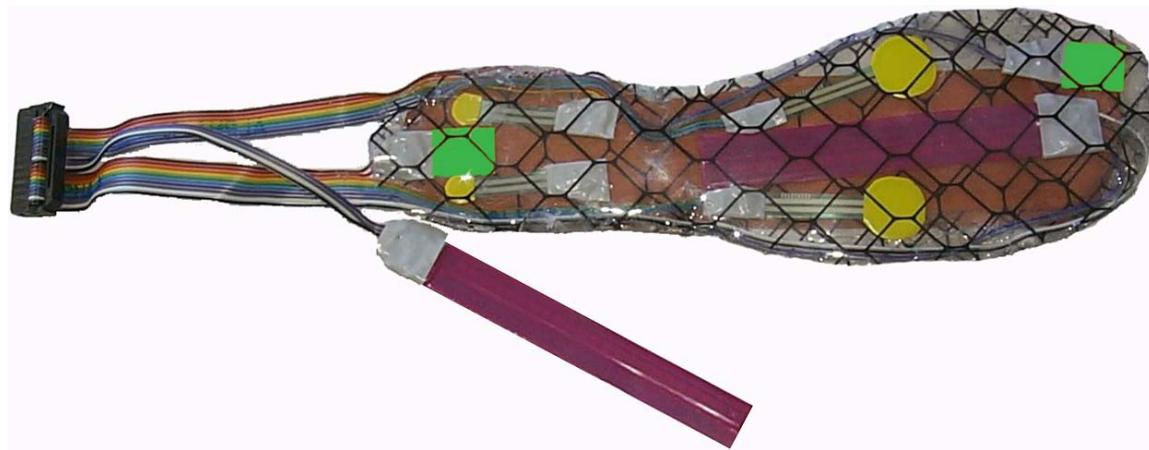


Figure 3.8 Photograph of an insole sensor

The second bi-directional bend sensor was connected to the electronics via the same ribbon cable. It was placed between the back of the shoe and the shin, and attached to the ankle with a nylon ankle-strap, as shown in Figure 3.9. The nylon ankle-strap was fitted around the subject's ankle, and held the bend sensor against the ankle, so to minimize buckling.



Figure 3.9 Bend sensor in ankle strap

When the electric field sensor was used, a ground plane was placed on top of the insole, and the driven shield of the electric field sensor was placed on the bottom of the insole; the sensors of the electric field sensor were attached to the bottom of the shoe. The ground plane, driven shield and electric field sensors used the smaller header to connect, as described in Section 3.2.1; a cross section of the shoe showing the layout of all these parts is shown in Figure 3.10.

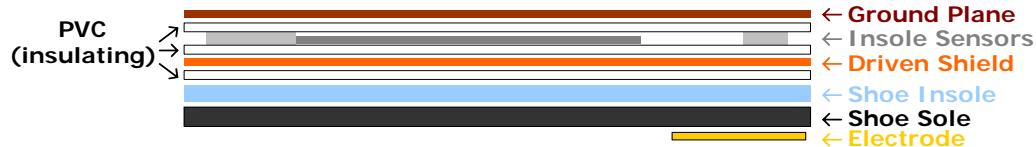


Figure 3.10 Cross section showing the electric field sensor components

To protect the insole sensors from the humid conditions of the shoe, the sensors were enclosed between two 0.02 inch sheets of clear Type 1 polyvinyl chloride (PVC) heavy duty film sheets. PVC sheets are rated to withstand humid and warm environments for up to three years. Insoles could be made in multiple sizes; for testing, one size was used, and sensors were moved to adjust for each subject's foot size.

3.2.3 Shoe Attachment

A shoe attachment was designed to hold the stack of printed circuit boards, the power board, and the antenna. It was designed such that the bulk of the volume was located behind the heel, so as to have a minimum effect on the gait. The attachment was made

from 0.125 inch polyethylene terephthalate glycol (PTG) sheets. PTG is thermoformable, machinable, and shatter-resistant.

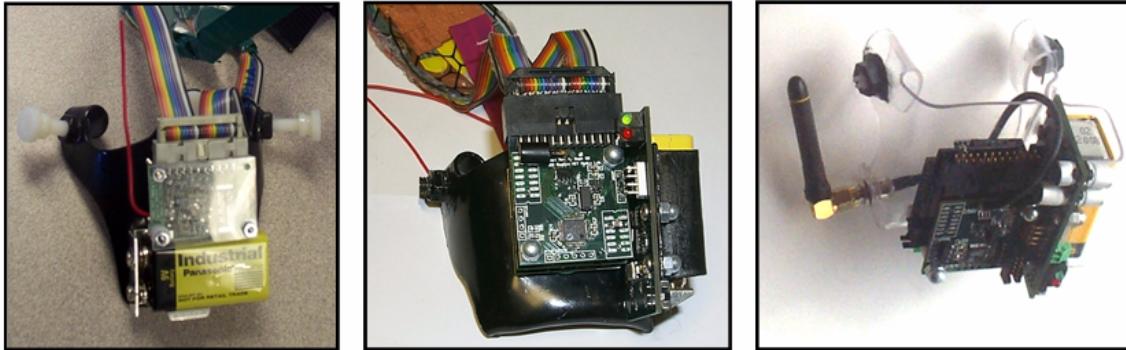


Figure 3.11 Photos of three versions of the shoe attachment

Three versions of the shoe attachment, shown in Figure 3.11, were designed, both to accommodate changes in the stack hardware and to improve the mechanical attachment. On the far left is the first prototype, which had three large screws at the top to fasten the attachment to the shoe. The middle photo is the second prototype, which had two screws at the left and the right, and the power board was rotated to the side to decrease the length of the attachment. The final prototype featured longer plastic hooks at the location of the two set screws to help keep the attachment stable on the shoe, included a mounting point for the antenna, and improved alignment between the main stack and the power board. Both the second prototype and the final prototype had a plastic loop at the bottom of the attachment, through which fishing line was threaded, looped under the shoe, and tied to the laces, to reduce motion of the attachment during heel strike. The left and right attachments of the final prototype are shown in Figure 3.12 (directions for fabricating the attachments are in Appendix D.3).

The attachment was designed for walking shoes. The plastic loops pulled the flexible shoe material of the walking shoes away from the foot, so that the plastic loops did not rub against the foot. The top of the shoe attachment was at the same height as the back of the shoe, which helped to keep the shin from hitting the electronics while walking. As most walking shoes have a large heel, the height of the shoe attachment was short in relation to the shoe height, which resulted in a large clearance between the bottom of the shoe attach-

ment and the floor, and helped to keep the attachment from hitting the floor while walking. The mounting points for the antennas were on the outside of each foot, in order to not affect the clearance between the two feet.

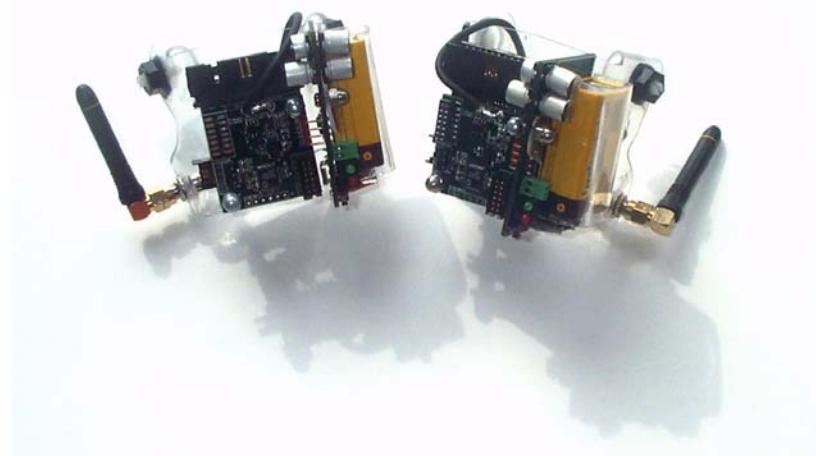


Figure 3.12 Final prototype of the GaitShoe attachment

The attachments are shown on a pair of shoes in Figure 3.13. The fishing line which held the bottom of the attachment against the shoe and was tied through the eyelets is visible on the side of the shoes. The co-axial cable connected to the electric field sensors on the bottom of the shoe is visible on the left shoe.



Figure 3.13 Photo of the GaitShoe system on two shoes

The fishing line helped to diminish vibration of the shoe attachment as a result of heel strike. Figure 3.13 shows the output of the accelerometers when the heel of an empty shoe is hit against the floor. The accelerometers demonstrate an excitation frequency around 20 Hz, which is quickly damped.

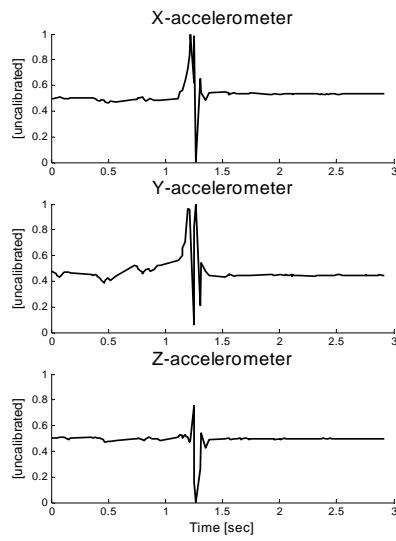


Figure 3.14 Accelerometer response to impact at heel

3.2.4 The Basestation

The final component of the GaitShoe system was the basestation. The basestation consisted of a metal box, with an antenna mounted externally. Inside the metal box was a main board, a power board, and a programming board (described in Appendix D.1.5) containing a MAX233 serial converter chip, to send the data to a laptop or desktop computer via the serial cable. The basestation is shown in Figure 3.15; on the outside of the box are a power switch, indicator lights, and a female DB-9 connector for the serial cable. In addition, a BNC connector provided a method of input from the MGH Biomotion Lab equipment via an optoisolated trigger connected to an input pin on the microcontroller.



Figure 3.15 Photo of the basestation

3.3 Sensor Specifications

This section discusses each sensor, including an overview of the working principle of the sensor, and the signal conditioning and implementation used in the GaitShoe system. Schematics of each sensor implementation are included and analyzed, and a table summarizing relevant parameters of each sensor (as provided by the manufacturers) is included; sensitivities, zero offsets, and cutoff frequencies refer to the output after any conditioning electronics shown in the schematics.

Bandwidth Requirements

The average step rate of adults is just under 120 steps per minute, which corresponds to a stride frequency of 1 Hz. The harmonic content of seven leg and foot markers was analyzed, and 99.7% of the signal power was found to be contained below 6 Hz [67]. In addition, it has been shown that if data is collected at 24 Hz, there will be negligible errors in the kinetic and energy analyses for normal gait speeds [68]; however, for kinematic analysis, a higher sampling rate, such as 50 Hz or above, may be necessary to capture all the information [69]. Thus, the sensors in the GaitShoe were all selected and set such that the low-pass cutoff frequencies (the 3 dB bandwidths) were greater than 25 Hz.

3.3.1 Accelerometer

The accelerometer selected for use in this application was the ADXL202E, a dual axis linear accelerometer from Analog Devices. The ADXL202E is a small, low-power, micro-electro-mechanical system (MEMS) accelerometer, and acceleration measurements have a full-scale range of $\pm 2g$.

Two of these dual-axis accelerometers were used, oriented¹ perpendicularly to each other. This resulted in a single measurement along each of two axes, and in duplicate measure-

1. The relative orientation between the two accelerometers, in the plane defined by the axis in one accelerometer corresponding to the vertical axis, and the axis in the second accelerometer corresponding to the horizontal axis, was determined using the gravitational vector, as described in Section 4.4.2.

ments along the third axis; one of the measurements along the duplicate axis was discarded.

Working Principle

The ADXL202E is constructed using a surface micro-machining process. The proof mass is suspended above the substrate by polysilicon springs, and can move in two perpendicular axes. Along each of the four sides of the square proof mass are eight sets of fingers which are positioned between plates affixed to the substrate; each finger and pair of plates results in a differential capacitor. When subjected to acceleration, the proof mass moves from its neutral position. Changes in the differential capacitances correspond to deflection of the proof mass due to acceleration [70].

This type of accelerometer measures both dynamic acceleration (resulting from shock, vibration, linear motion, or other types of motion), and static acceleration (resulting from gravity). Thus, for analysis of acceleration due only to linear motion, the orientation of the accelerometer must be determined so that the gravity contribution can be subtracted from the total output.

Implementation

The ADXL202E provides the acceleration output as either a digital duty-cycle or an analog voltage. The analog output was used in the final design¹ of the IMU board. Figure 3.16 shows the schematic of the circuitry used for one of the accelerometers, Figure 3.17 shows the placement of the components for both ADXL202E sensors on the IMU board, and Table 3.2 lists relevant parameters for this implementation. The bandwidth was limited externally by adding 47 nF capacitors, C_1 and C_2 , at the X_{filt} and Y_{filt} pins, resulting in a cutoff frequency of 100 Hz. A 0.1 μF capacitor, C_3 , was placed between the V_{dd} pin and ground to decouple the 3.3 V power supply. A ferrite bead, R_6 , was placed between the

1. An earlier implementation used the duty-cycle output, but timing the duty-cycle with the microcontroller was very time-intensive (the shortest duty-cycle available was 1 ms), so the analog output was used in the final version to enable fast data acquisition (the other sensor outputs are all analog, as well).

power supply and the V_{dd} pin to further reduce digital noise (as recommended when a microcontroller shares the power supply; a 100Ω resistor can also be used if a ferrite bead is unavailable). The resistor, R_5 , at pin 2 was provided for the duty cycle converter and was set to $125\text{ k}\Omega$ (a resistor under $10\text{ M}\Omega$ is recommended to keep the duty cycle converter running, even when the analog output is used).

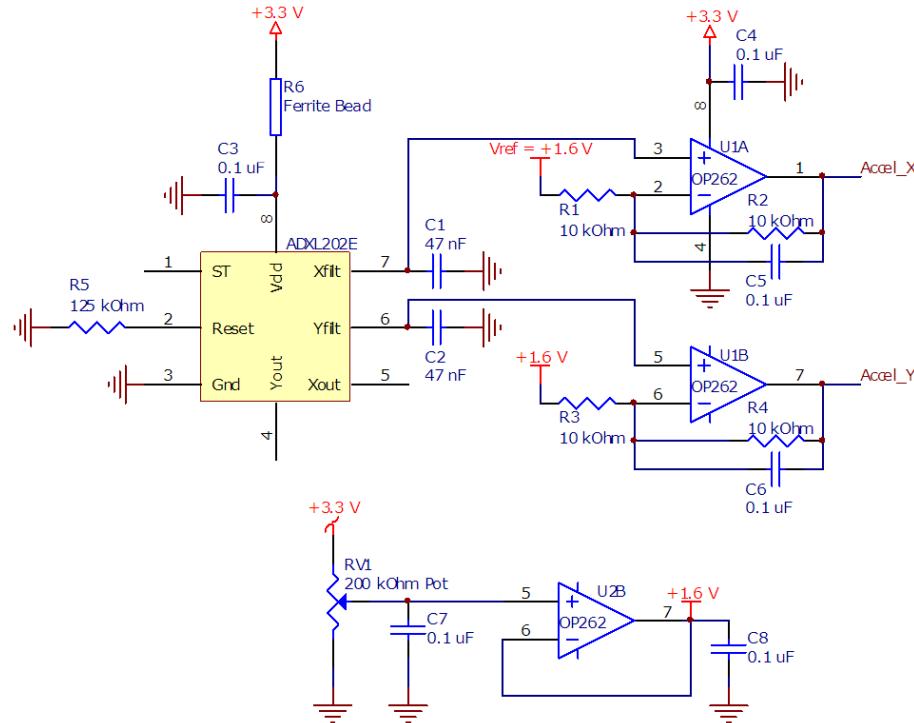


Figure 3.16 Schematic of the ADXL202E

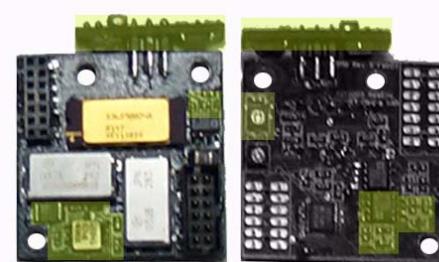


Figure 3.17 Photo of the IMU board, with the ADXL202E components highlighted

The X_{filt} and Y_{filt} pins have an output impedance of $32\text{ k}\Omega$; since they cannot directly drive a load, the output of each pin was buffered by a non-inverting op-amp follower. The relationship between the voltage, V_{filt} , on pin X_{filt} , and the output voltage from the op-amp, V_{out} , is described by Eq. 3.1, where V_{ref} is the voltage supplied to the inverting input

through R_1 . The two resistors, R_1 and R_2 , were both set to $10\text{ k}\Omega$ for a gain of 2. The reference voltage, V_{ref} , was set to 1.6V, in order to keep the output centered around 1.6 V. The 1.6 V was provided through use of a voltage divider connected to the 3.3 V power supply; the 3.3 V was divided via a $200\text{ k}\Omega$ potentiometer, RV_1 , and buffered by a non-inverting op-amp with a gain of 1.

$$V_{out} = \left(1 + \frac{R_2}{R_1}\right) \cdot (V_{filt} - V_{ref}) + V_{ref} \quad (3.1)$$

Capacitor C_5 , in parallel with R_2 , reduces extraneous noise, with a cutoff frequency of 160 Hz. The same is true for the output from pin Y_{filt} . Finally, the power pin on the op-amp has a $0.1\text{ }\mu\text{F}$ capacitor, C_4 , to minimize noise from the power supply; the voltage divider which provides the 1.6 V also has two $0.1\text{ }\mu\text{F}$ capacitors, C_7 and C_8 , to reduce noise in the supply line [71].

TABLE 3.2 Relevant parameters of the Analog Devices ADXL202E accelerometer [71]

Parameter	Value
Measurement range	$\pm 2g$
Sensitivity	Typical: 377.5 mV/g , min: 313.0 mV/g , max: 452.5 mV/g
Zero offset	Typical: 1.65 V , min: 1.0 V , max: 2.3 V
Cutoff frequency	100 Hz
Resonant frequency	10 kHz
Nonlinearity	0.2% full scale
Temperature sensitivity	$2.0\text{ mg/}^{\circ}\text{C}$ from 25°C
Noise floor	Typical: $200\text{ }\mu\text{g}/\sqrt{\text{Hz}}$ rms, max: $1000\text{ }\mu\text{g}/\sqrt{\text{Hz}}$ rms
RMS noise	Typical: 2.5 mg
Peak-to-peak noise estimate	95% probability: 10.1 mg
Quiescent supply current	Typical: 0.6 mA , Max: 1.0 mA
Power draw	Typical: 1.98 mW
Shock survival	Up to 1000g while unpowered, up to 500g while powered
Package size	$5\text{ mm} \times 5\text{ mm}$, 2 mm tall
Package weight	< 1.0 grams

3.3.2 Gyroscopes

Two types of MEMS vibrating rate gyroscopes were used in this application: the Murata ENC-03J gyroscope, and the Analog Devices ADXRS150 gyroscope. The ADXRS150 is a yaw gyroscope, meaning it measures rotation about the axis perpendicular to the plane of the sensor. In contrast, the ENC-03J measures rotation about the long axis in the plane of the sensor. In order to measure rotation about three axes on one flat circuit board, two ENC-03J gyroscopes were placed nominally perpendicularly¹ to each other, with the ADXRS150 placed in the same plane.

Working Principle

Both types of gyroscopes measure angular velocity through use of a vibrating element and the Coriolis effect. When rotation is applied to a vibrating element, conservation of angular momentum results in a secondary oscillation orthogonal to both the axis of vibration and the axis of rotation (this is the Coriolis effect). The magnitude of this secondary oscillation is proportional to the magnitude of the rate of rotation [72].

In the Murata ENC-03J, the vibrating element is a piezoelectric ceramic prism, which is supported such that it can move freely within the plane normal to its long axis. The prism is driven at its resonant frequency by a piezoelectric transducer. When rotation occurs about the long axis of the prism, the resulting secondary oscillation is measured by a second piezoelectric transducer. This type of system is susceptible to external vibrations. To minimize resonant coupling when two of these sensors are placed in close proximity to each other, Murata sells an "A" and a "B" version, which have slightly different resonant frequencies [73].

In the Analog Devices ADXRS150, the vibrating element is a polysilicon structure. This structure consists of a mass tethered to an inner frame, which is tethered to the substrate. The mass is allowed to move freely only along the axis of vibration, and the inner frame is

1. The relative orientation between the gyroscopes was not explicitly determined.

allowed to move freely only along the axis normal to the axis of vibration. The inner frame has fingers positioned between plates affixed to the substrate. When rotation occurs about the axis orthogonal to the plane of the polysilicon structure, the resulting secondary oscillation between the inner frame and the substrate can be measured by the changes in capacitance between the fingers on the inner frame and the plates on the substrate. In addition, the ADXRS150 has two of these structures, placed adjacent to each other, but operating in anti-phase. By examining the differential signal from the two structures, the conditioning electronics contained within the chip are able to reject common-mode signals unrelated to the angular rate, such as external shocks and vibrations [74] [75].

Implementation of the Murata ENC-03J Gyroscope

Two Murata ENC-03J sensors were used, one type A and one type B. The ENC-03J sensor is a dual-inline-pins (DIP) style package, and has been replaced by ENC-03M which is a surface-mount part. The Murata data sheets indicate that the ENC-03M is a direct replacement for the ENC-03J, with the same specifications other than size and weight (the hardware developed for this thesis includes a footprint for the ENC-03M within the footprint for the ENC-03J, as discussed in Appendix D.1.1; the ENC-03J was used in all testing).

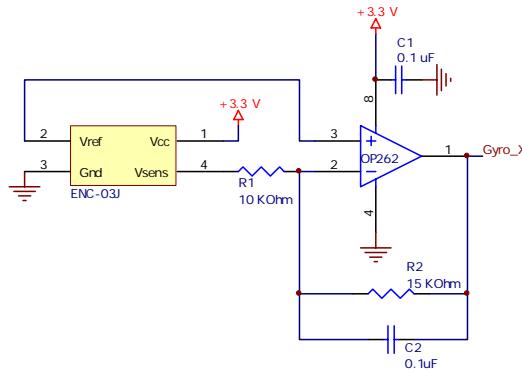


Figure 3.18 Schematic of the ENC-03J

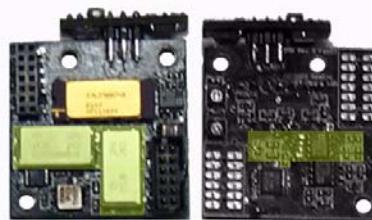


Figure 3.19 Photo of the IMU board, with the Murata ENC-03J components highlighted

The schematic for the implementation of one of the ENC-03J sensors is shown in Figure 3.18. Figure 3.19 shows the placement of the components for both ENC-03J sensors on the IMU board, and the relevant parameters are listed in Table 3.3. The output was buffered by an op-amp follower, in an inverting configuration. The relationship between the output voltage of the sensor, V_{sens} , and the output voltage from the op-amp, V_{out} , is described by Eq. 3.2, where V_{ref} is the voltage supplied to the inverting input through R_1 . While the op-amp was used in the inverting configuration, the output was always positive since V_{ref} , the zero-offset of the sensor provided by the ENC-03J, kept the output centered. Resistor R_1 was set to 10 kΩ and resistor R_2 was set to 15 kΩ, for a gain of 1.5. Capacitor C_2 , in parallel with R_2 , reduced extraneous noise, with a cutoff frequency of 106 Hz (the sensor itself has a cutoff frequency of 50 Hz). The power pin on the op-amp had a 0.1 μF capacitor, C_1 , to minimize noise from the power supply [73].

$$V_{out} = -\frac{R_2}{R_1} \cdot (V_{sens} - V_{ref}) + V_{ref} \quad (3.2)$$

TABLE 3.3 Relevant parameters of the Murata ENC-03J gyroscope [73]

Parameter	Value
Measurement range	± 300°/sec
Sensitivity	0.37 mV/°/sec
Zero offset	1.65 V
Cutoff frequency	50 Hz
Nonlinearity	± 5% full scale
Temperature sensitivity	± 20% at -5 or +75°C
Current consumption	Max: 5 mA
Power draw	Max: 16.3 mW
Package size	15.44 mm x 8 mm, 4.3 mm tall (not including pins)
Package weight	<1.0 grams

Implementation of the Analog Devices Gyroscope

The Analog Devices ADXRS150 used in the GaitShoe system was an early demo version, and was a sixteen pin DIP package. The ADXRS150 is now available for purchase as a surface mount part, and has nearly identical specifications to the demo part used, other than size and weight (Appendix D.1.1 shows the pin-to-pin mappings between the two versions). All specifications and operating parameters in this chapter correspond to the demo version.

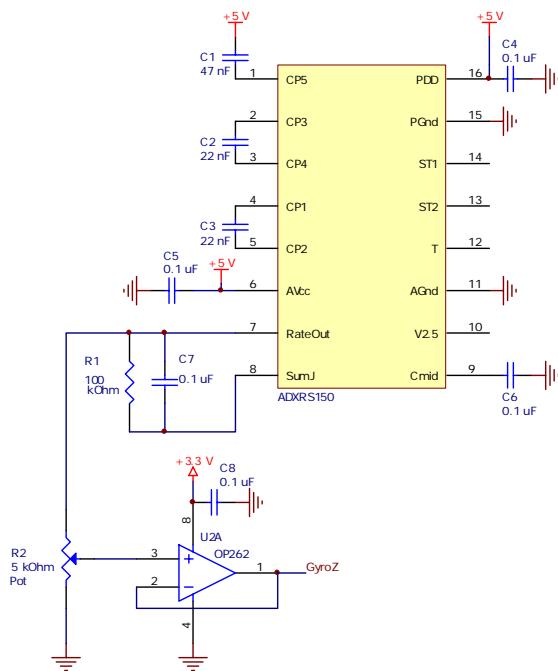


Figure 3.20 Schematic of the ADXRS150



Figure 3.21 Photo of the IMU board, with the ADXRS150 components highlighted

The implementation of the ADXRS150 is shown in Figure 3.20, Figure 3.21 shows the placement of the components for the ADXRS150 on the IMU board, and Table 3.4 lists the relevant parameters. The ADXRS150 required a 5 V power supply, and was the only

sensor in the system to use 5 V. Capacitors C_1 , C_2 , and C_3 were set as specified (22 nF, 22 nF, and 47 nF, respectively) in the data sheet, and had to be located close to their respective pins, as these capacitors were used in the charge pump that provided 16 V for the electrostatic resonator. Capacitors C_4 and C_5 were each set to 0.1 μ F, as specified, and also needed to be placed as near their respective pins as possible; these capacitors were used to minimize noise injection from the charge pump into the supply voltage. Capacitor C_6 was set to 0.1 μ F, as recommended, to set the cutoff frequency before the final amplification stage to 400 Hz ($\pm 35\%$, due to tolerances of two internal resistors). This low pass filter served to eliminate high frequency artifacts prior to the final amplification. Capacitor C_7 was set to 0.1 μ F, and resistor R_1 was set to 100 k Ω , in order to set the bandwidth and adjust the measurement range of the sensor. Resistor R_1 was in parallel with an internal resistor, $R_{out_internal}$, resulting in an effective $R_{out} = 64.3$ k Ω . This resulted in a cutoff frequency for this sensor of 25 Hz. By reducing R_{out} by approximately a third (as compared to $R_{out_internal}$), the sensitivity was decreased by approximately a third, and the measurement range was increased by approximately a third. The output of the sensor was divided via a potentiometer, R_2 , such that the resulting output was centered around 1.65 V, with a full-scale range from 0 - 3.3 V. Finally, the output was buffered by a non-inverting op-amp follower with a gain of 1, and capacitor C_8 was set to 0.1 μ F to reduce noise into the op-amp from the power supply [75].

3.3.3 Force Sensitive Resistors

Two sizes of force sensitive resistors (FSRs) manufactured by Interlink Electronics were used, FSR-402, which has a circular sensing area with a diameter of 12.7 mm, and FSR-400, which has a circular sensing area with a diameter of 5 mm. Two of the FSR-400 sensors were placed underneath the heel pad, one medially and the other laterally. One of the FSR-402 sensors was placed underneath the first metatarsal head, and a second FSR-402 sensor was placed underneath the fifth metatarsal head.

TABLE 3.4 Relevant parameters of the Analog Devices ADXRS150 gyroscope [75]

Parameter	Value
Measurement range	$\pm 420 \text{ }^{\circ}/\text{sec}$
Sensitivity	Typical 2.9 mV/ $^{\circ}/\text{sec}$, min: 2.8 mV/ $^{\circ}/\text{sec}$, max: 3.2 mV/ $^{\circ}/\text{sec}$
Zero offset	Typical: 1.65 V, min: 1.4 V, max: 1.8 V
Cutoff frequency	25 Hz
Resonant frequency	16 kHz
Nonlinearity	0.1% full scale
Temperature sensitivity ^a	Min: 4.2 mV/ $^{\circ}/\text{sec}$, max: 4.9 mV/ $^{\circ}/\text{sec}$
Rate noise density	$0.05 \text{ }^{\circ}/\text{sec} / \sqrt{\text{Hz}}$
Quiescent supply current	Typical: 5.0 mA, Max: 8.0 mA
Power draw	Typical: 25 mW
Shock survival	Up to 30,000g while unpowered, up to 500g while powered

a. Greatest deviations from initial value at 25°C to worst case value at T_{min} or T_{max}.

Working Principle

A force sensitive resistor is a type of sensor which experiences a decrease in electrical resistance when force is applied orthogonally to the active area of the sensor. Though less accurate than a load cell, FSRs are generally inexpensive, and when manufactured from polymers, the typical thickness is on the order of 0.25 mm [76].

The FSRs manufactured by Interlink Electronics are polymer-based sensors, and consist of three layers. The lowest layer is a flexible substrate which has been coated with a printable semi-conductor material. The middle layer is a spacer adhesive, with material only along the outline of the part, providing an open region at the active area of the device. The top layer is a flexible substrate, printed with interdigitating electrodes and two printed leads which connect to solder tabs. The active area of the sensor is the area containing the electrodes. FSR-400 uses a 0.10 mm layer of polyethersulfone for the top and bottom layers, with a 0.05 mm layer of acrylic for the spacer. FSR-402 uses a 0.13 mm layer of polyetherimide for the top and bottom layers, with a 0.15 mm layer of acrylic for the spacer. Polyethersulfone is a transparent substrate which has excellent temperature resistance,

moderate chemical resistance, and good flexibility. Polyetherimide, on the other hand, is a semi-transparent substrate which has excellent temperature resistance, excellent chemical resistance, and limited flexibility [77].

The semi-conductor material on the lowest layer provides an electrical connection between the sets of interdigitating electrodes. Without any force applied, the adhesive provides an air gap between the semi-conductor and the electrodes, so the resistance across the sensor is high. When force is applied across the active area, the electrodes are pressed into the semi-conductor, which reduces the resistance across the sensor.

If the sensor is bent, local changes along the line of bending may result in a decrease in resistance, which would cause a false reading of applied force. Changes in temperature and humidity can also affect the output of the FSR.

Implementation

A voltage divider was used to measure the change in resistance of the FSRs. The circuit implemented is shown in Figure 3.22. Figure 3.23 shows the placement of the components for all four FSRs on the Tactile board, and relevant parameters of the FSRs used are listed in Table 3.5. The same values of resistors were used for both the FSR-400 and the FSR-402 sensors. To lower the impedance of the signal, the output of the voltage divider was followed by a bipolar junction transistor (BJT), in the typical *n*p*n* configuration, used as an emitter follower; there was a 0.6 V voltage drop across the BJT. As shown, a 1 k Ω resistor, R_1 , was used in the voltage divider, and a 10 k Ω resistor, R_2 , was placed between the emitter and the ground. The relationship between the FSR output, V_{out} , and the resistance of the FSR, R_{FSR} , is described by Eq. , where R_1 is 1 k Ω and V_{cc} is 3.3 V.

$$V_{out} = \left(\frac{R_1 + R_{FSR}}{R_{FSR}} \right) \cdot V_{cc} - 0.6V \quad (3.3)$$

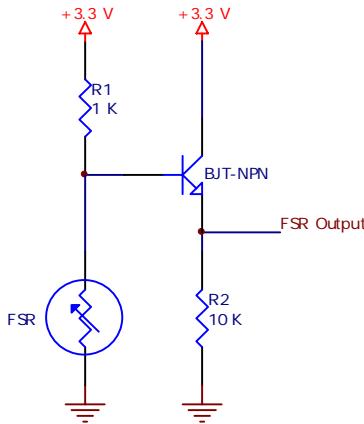


Figure 3.22 Schematic of the FSR

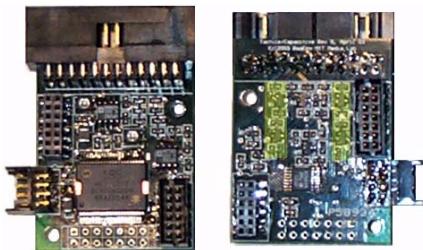


Figure 3.23 Photo of the Tactile board, with the FSR components highlighted

TABLE 3.5 Relevant parameters of the Interlink FSR-400 and FSR-402 [77]

Parameter	Value
Force sensitivity range	<100 g to >10 kg, depending on mechanics
Pressure sensitivity range	<0.1 kg/cm ² to >10 kg/cm ² , depending on mechanics
Part-to-part repeatability	±15% to ±25% of established nominal resistance ^a
Single part repeatability	±2% to ±5% of established nominal resistance ^a
Cutoff frequency	500 Hz
Device rise time	1-2 msec
Resolution	0.5% full scale
Current consumption	FSR-400: 0.2 mA, FSR-402: 1.3 mA
Power draw	FSR-400: 0.66 mW, FSR-402: 4.3 mW
Lifetime	>10 million actuations
Package size	FSR-400: 7.5 mm x 38.1 mm, 0.30 mm thickness FSR-402: 18.3 mm x 54.1 mm, 0.46 mm thickness

a. With a repeatable actuation system.

3.3.4 Bend Sensors

The bend sensors used were the FLX-01, manufactured by The Images Company. Each FLX-01 is 11.4 cm long and 0.64 cm wide, with a thickness of 0.5 mm. One bi-directional bend sensor was located in the insole to measure the flexion at the metatarsal-phalangeal joint, and the other was placed between the shin and the back of the shoe to measure the dorsi-/plantar- flexion of the foot.

Working Principle

The bend sensors work in a manner similar to the FSRs: the resistance through the sensor changes as the sensor is actuated. However, instead of having interdigitating electrodes, the electrodes are printed linearly along the length of the sensor. When the bend sensor is bent in the direction which lengthens the spacing between the printed electrodes, the resistance increases (bending in the direction which shortens the spacing has a negligible effect on the resistance). When unbent at 0° , the nominal resistance of the sensor is $10\text{ K}\Omega$, and when bent to 90° in the direction of sensitivity, the resistance is on the order of $30\text{-}40\text{ K}\Omega$. Since each individual bend sensor measures bend in only one direction, two sensors were used in pairs, such that their sensitive bending directions would complement each other, resulting in a measuring range of $\pm 180^\circ$, as shown in Figure 3.24. A differential circuit was used to combine the outputs, in order to provide an output corresponding to the bi-directional bend.



Figure 3.24 Bend sensors shown back to back

Implementation

The differential circuit implemented to combine the individual uni-directional bend sensor outputs into a bi-directional bend output is shown in Figure 3.25, Figure 3.26 shows the placement of the components for both bi-directional bend sensors on the Tactile board, and relevant parameters of the FLX-01 are listed in Table 3.6. A voltage divider was used

to measure the change in resistance of each individual bend sensor, with resistors R_1 and R_2 each set to $20\text{ k}\Omega$. The outputs from each voltage divider were then combined through an op-amp set as a differential amplifier. The relationship between the bi-directional bend output, V_{out} , and the resistance of the bend sensors, R_A and R_B , is described by Eq. 3.4, where R_1 is $10\text{ k}\Omega$ and V_{cc} is 3.3 V (note $R_3 = R_1$, $R_4 = R_2$, and $R_7 = R_8$). Since R_2 is $100\text{ k}\Omega$ and R_8 is $220\text{ k}\Omega$, there is a gain of 2.2 across the op-amp. Assuming the bend sensors are well-matched when flat, the output is centered around 1.65 V, with the output greater than 1.65 V when bend sensor B is bent in its sensitive direction, and less than 1.65 V when bend sensor A is bent in its sensitive direction. Capacitor C_1 , in parallel with R_3 , reduces extraneous noise, with a cutoff frequency of 154 Hz.

$$V_{out} = \frac{R_8}{R_2} \cdot \left(\frac{R_B}{R_B + R_1} - \frac{R_A}{R_A + R_1} \right) \cdot V_{cc} + \frac{V_{cc}}{2} \quad (3.4)$$

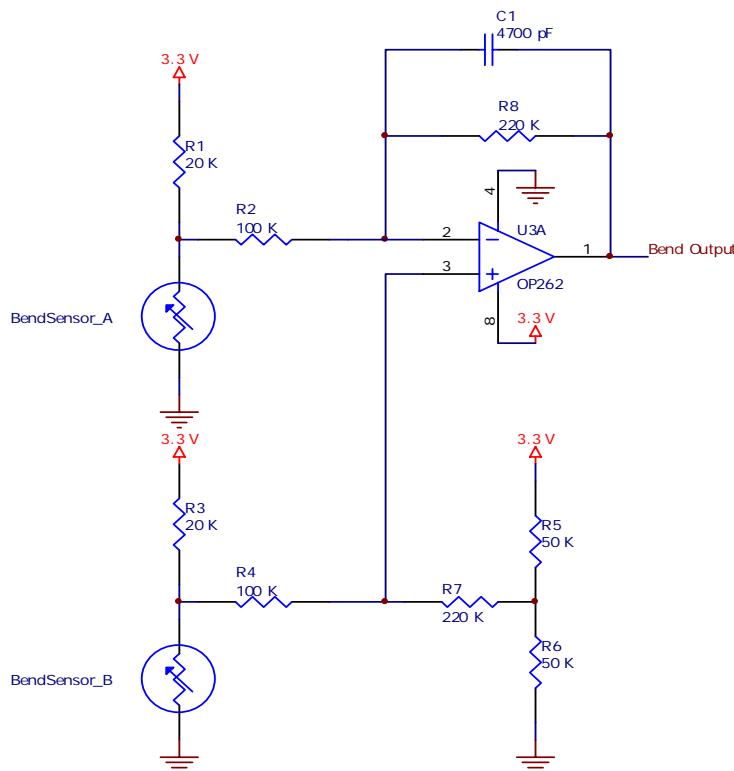


Figure 3.25 Schematic of the bi-directional bend sensor

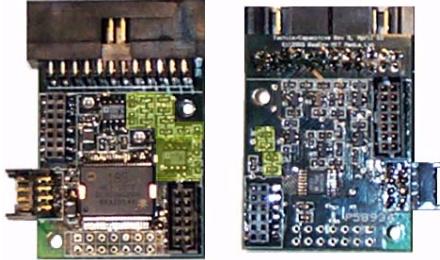


Figure 3.26 Photo of the Tactile board, with the bi-directional bend sensor components highlighted

TABLE 3.6 Relevant parameters of The Images Co. FLX-01 [78]

Parameter	Value
Angle sensitivity range	0° to 90°
Cutoff frequency	154 Hz
Package size	11.4 cm x 0.64 cm, 0.5 mm thickness

3.3.5 Polyvinylidene Fluoride Strips

Two polyvinylidene fluoride strips, part LDT0 made by Measurement Specialities, were used to measure the dynamic force applied across the sensor. One was placed under the heel and the other was placed under the great toe.

Working Principle

Polyvinylidene fluoride (PVDF) is a manufactured polymer that can be poled to become a piezoelectric material. Piezoelectric materials have somewhat of a crystalline structure (PVDF is a semi-crystalline homopolymer) which generates an charge difference when subjected to stress. While a piezoelectric material is electrically neutral at rest, stress deforms the crystalline structure such that the dipoles align and a polarity develops along one axis, resulting in a net charge expressed across the structure; this is a reversible physical phenomenon [79].

The LDT0 has a thin film of PVDF laminated between two electrodes; forces applied to this sensor result in stress across the piezoelectric material. The electrical equivalent of this sensor is a voltage source in series with a capacitor, with a shunting leakage resistor.

The capacitance of the LDT0 is on the order of 400 pF, and the leakage resistor is very large (on the order of 10^{12} - 10^{14} Ω), which means that the sensor has a very high output impedance at low frequency. Piezoelectric sensors are typically interfaced with matching charge amplifiers, or voltage amplifiers with high input resistances [76] [80].

Implementation

The circuit implemented for the PVDF strips provides an output corresponding to dynamic force, and is shown in Figure 3.27. Figure 3.28 shows the placement of the components for both PVDF strips on the Tactile board, and relevant parameters of the PVDF strips used are listed in Table 3.7.

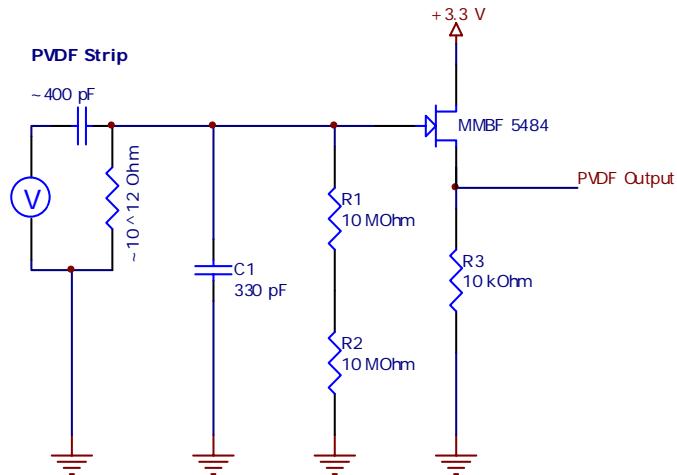


Figure 3.27 Schematic of the PVDF strips

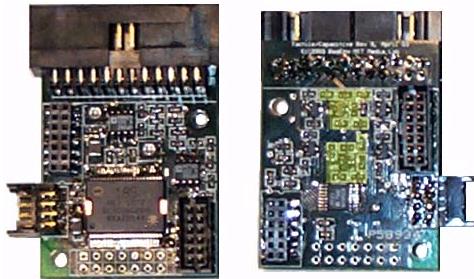


Figure 3.28 Photo of the Tactile board, with the PVDF strip components highlighted

The output from the PVDF strip was connected to the gate of an *n*-channel junction field-effect transistor (JFET). The JFET was used as a source follower, with a source resistor, R_3 , set to 10 kΩ; JFETs have extremely low leakage current, which results in a high input

impedance, as necessary for buffering the PVDF strip. When the PVDF strip was bent by a force, it generated a voltage difference across its capacitance, which biased the JFET, and caused current to flow from the drain to the source, proportionally increasing the voltage at the "PVDF output," as labeled in Figure 3.27. The PVDF strip was shunted by two resistors in series, R_1 and R_2 , each set to $10 \text{ M}\Omega$ for a total¹ resistance of $20 \text{ M}\Omega$. This shunt attenuated low frequency drift from charge buildup on the piezo strip, and this attenuation resulted in an output signal that corresponded to dynamic changes in the force applied across the sensor. The (first-order, high-pass) cutoff frequency for the system was 20 Hz, and was determined by resistors R_1 and R_2 ($20 \text{ M}\Omega$ total) and the LDT0 capacitor ($\sim 400 \text{ pF}$). The shunt capacitor, C_1 (330 pF), provided a frequency-independent attenuation of the piezo's signal.

TABLE 3.7 Relevant parameters of the Measurement Specialties LDT0 PVDF strip [80]

Parameter	Value
High-pass cutoff frequency	20 Hz
Package size	25.0 mm x 13.0 mm, 0.15 mm thickness

3.3.6 Electric Field Sensor

An electric field sensor was developed during the latter part of the project, using an electric field imaging device manufactured by Motorola, part MC33794DH, for occupant detection in automotive seat applications [81]. This sensor was used in testing to measure the height of the heel above floor via capacitive coupling. Another electrode was later configured to measure the height of the toes/metatarsals above the floor.

Working Principle

An electric field sensor measures the capacitive coupling between an electrode at an oscillating potential and other electrodes or grounds [63]. When an object moves into the elec-

1. The use of two resistors allowed for more flexibility in setting the total resistance, as surface mount resistors in the high $\text{M}\Omega$ range were only available in a small number of discrete values.

tric field created by the oscillating electrode, it increases the capacitive loading of the electrode and changes the intrinsic capacitive coupling between the electrode and its environment. For measuring the height of the foot above the floor, both the transmitting electrode and the ground were placed on the bottom of the shoe. When the foot is "high" off the floor, all of the electric field lines emanating from the sensing electrode will shunt to ground via C_o . As depicted in Figure 3.29, the system (and the measurements presented in Section 4.8) consisted of two electrodes: an active sensing electrode partly surrounded by another electrode at local ground. When the foot is far off the floor, the capacitive loading is dominated by the intrinsic coupling between the sensing electrode and the ground electrode (C_o), as dictated by their geometry. As the foot nears the floor, however, the conductive and dielectric property of the flooring ($R_F \parallel C_F$) becomes increasingly significant, through the series chain of C_{FE} to $R_F \parallel C_F$ to C_{FG} and eventually dominates the intrinsic capacitance C_o when the foot is in contact with the floor. By measuring this change, the height of the foot above the floor can be estimated.

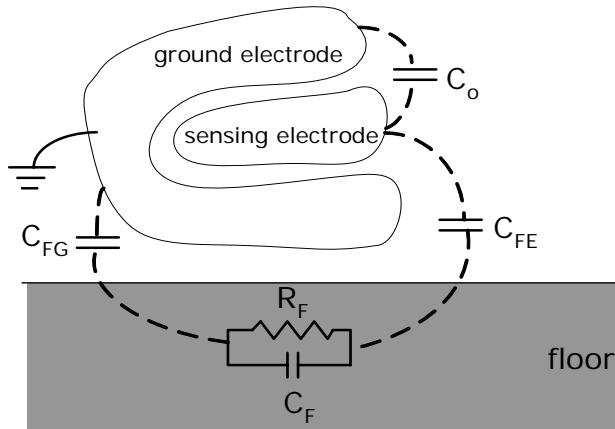


Figure 3.29 Schematic showing electrode coupling to both the ground and the floor

A second electrode configuration, developed after subject testing, consisted of circular electrodes on the bottom of the shoe, with a driven shield layer in the insole and a ground layer above the driven shield (the insole with the other sensors was located between the driven shield and the ground layer), as depicted in Figure 3.30. The driven shield effectively prevents coupling between the sensing electrode and the ground electrode, resulting in a larger measurement output range.

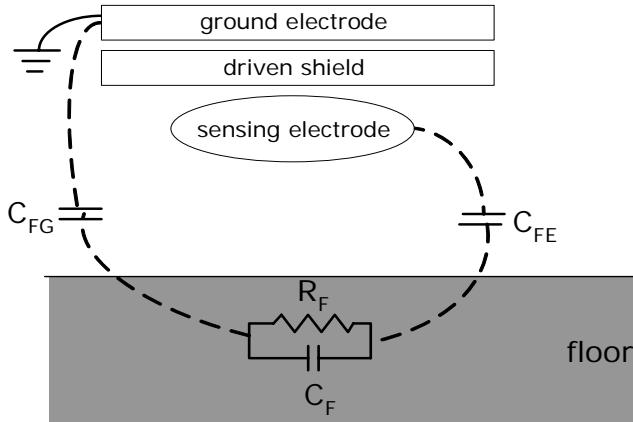


Figure 3.30 Schematic showing the second electrode configuration coupling to the floor

The MC33794DH drives the electrodes by generating a sine wave at 120 kHz, and measuring the capacitive coupling by detecting the load current. The sine wave has very low harmonic content, to reduce interference with other systems, and the frequency of the sine wave is adjustable via an external resistor. Up to nine electrodes can be driven by the MC33794DH; one electrode is selected at a time, by a 4 bit digital address, and the unselected electrodes are all grounded. To measure the capacitance on the selected electrode, the load current is converted to a DC level, filtered by an external capacitor, and multiplied and offset, resulting in an output range from 1.0 V to 4.0 V.

The MC33794DH also includes a shield driver, which provides a signal that follows the sinusoidal signal on the selected sensing electrode. This signal can be used to drive shielding electrodes that prevent coupling to nearby grounds. It can also be used to drive the shield on a co-axial cable connecting a sensing electrode. This effectively nulls the intrinsic capacitance of the co-axial cable, keeping measurement of the capacitance of the electrode minimally affected. In addition, a driven shield layer, constructed of an additional PVC insole lined with copper tape, and connected to the driven shield, was placed beneath the GaitShoe insole, to shield the sensing electrode from loading by the insole sensors.

Implementation

The circuit implemented for the MC33794DH is shown in Figure 3.31, Figure 3.32 shows the placement of the components on the Tactile board, and relevant parameters of the

MC33794DH are listed in Table 3.7. Due to its automotive legacy, this sensor required a +12 V input, and was the only component to require +12 V.

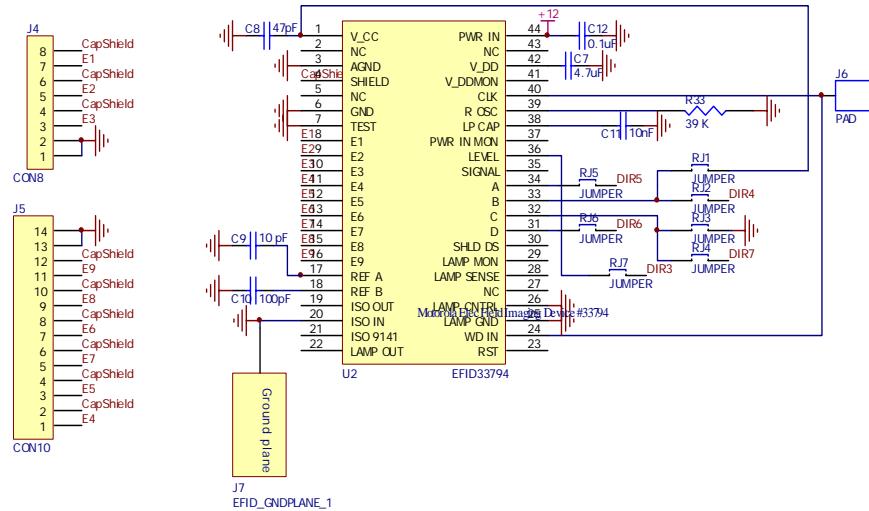


Figure 3.31 Schematic of the electric field sensor

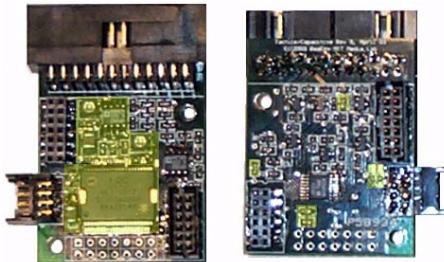


Figure 3.32 Photo of the Tactile board, with the electric field components highlighted

The configuration of the electrode, as used to measure the height above the floor for the last few subjects during subject testing, is shown in Figure 3.33. The capacitive tape along the perimeter of the shoe is connected to the ground of the +9 V battery, and the inner strip is connected to electrode E2 (the driven shield within the shoe shown in Figure 3.10 was not used with this electrode configuration). The second configuration, developed after subject testing, including two sensing electrodes, is shown Figure 3.34.

The various capacitors and resistors were set as recommended in the data sheet. Resistor R_{33} was set to $39\text{ k}\Omega$, which set the frequency of the sine wave to 120 kHz . Capacitor C_{11} was the external capacitor which filtered the receiver multiplexer signal, and was set to 10 nF , which allowed in a 99% settling time for the detector output in under 5.0 ms^1 .

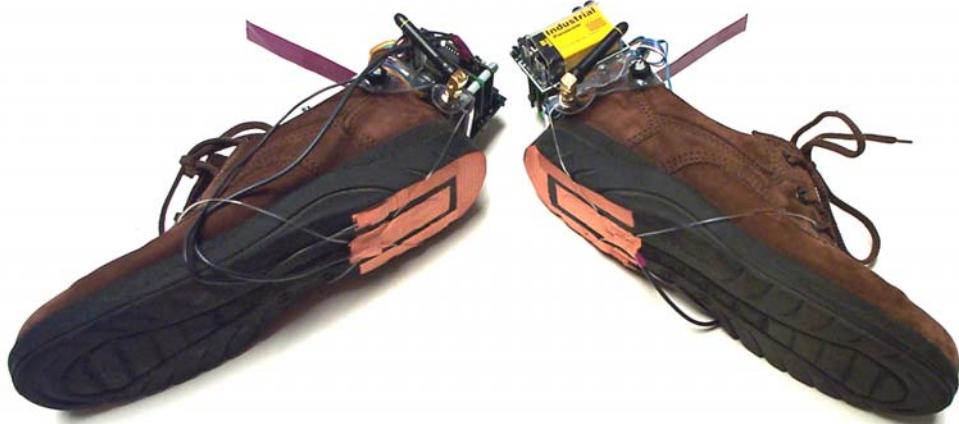


Figure 3.33 Example of an electrode set up on the bottom of a shoe to measure heel height

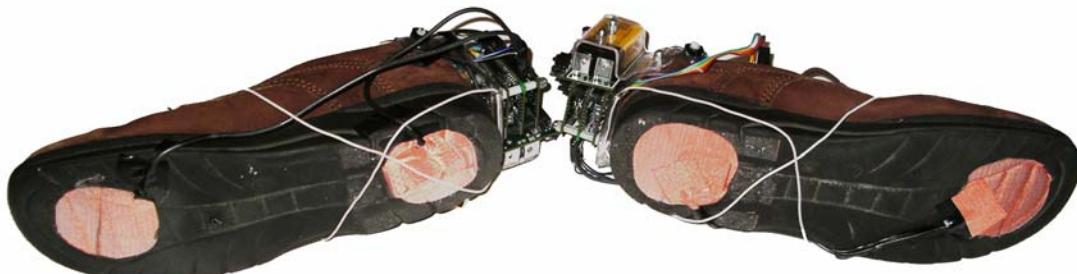


Figure 3.34 Example of two electrode set up on the bottom of a shoe to measure heel and toe height.

Capacitors C_9 and C_{10} were set to 10 pF and 100 pF respectively; these capacitors served as external reference capacitors, which allowed a method for determining the absolute capacitance on a selected electrode. Capacitor C_8 was set to 47 pF, C_{12} was set to 0.1 μ F, and C_7 was set to 4.7 μ F, all for filtering noise from the power lines. The "ground plane" pad connected to the circuit board ground at pin 20 was placed underneath the MC33794DH, and soldered to its heat sink. The heat sink was designed into this chip for its other applications (e.g., as a lamp driver); the capacitive sensing function consumes very little power, and does not actually require the heat sink.

1. All sensors on each shoe are sampled at 75 Hz, or, every 13.4 msec. The microcontroller turned each electrode on, and paused before measuring the output, in order to allow the output to settle.

Jumper pads RJ₁ - RJ₆ were used to allow the microcontroller to set the multiplexer and select the electrodes. To select two of the electrodes connected to the small receptacle shown in Figure 3.32 (e.g. for use with electrodes under the heel and the metatarsals), RJ₁ and RJ₃ were connected to pull each high and low via the local power supply and ground, respectively, RJ₂ and RJ₄ were left open, and RJ₅ and RJ₆ were connected to the microcontroller. Using the microcontroller, RJ₆ was pulled low, and RJ₅ was pulled low to select the first electrode, and high to select the second electrode. Jumper pad RJ₇ was connected to provide the output signal directly to the microcontroller.

The driven shield was used to drive the co-axial shield on the cables connecting the electrodes to the header pins. The driven shield was also used, in lieu of a ground plane, on the printed circuit board in the areas surrounding the header connections and the lines connecting the headers to the MC33794DH chip, to further shield the electric pins from any stray capacitance or noise on the board. In addition, the driven shield signal was routed to capacitive tape which formed a shield beneath the insole, to prevent any noise or load capacitance from the insole sensors from interfering with the sensing electrodes on the bottom of the shoe.

TABLE 3.8 Relevant parameters of the Motorola MC33794DH [82]

Parameter	Value
Measurement range	Typical: 10 pF - 100 pF
Sensitivity	Typical: 30 pF / V
99% settling time	Max: 5 ms
Cutoff frequency	Max: 200 Hz
Electric field frequency	120 kHz
Package size	14 mm x 16 mm; 3 mm tall

3.3.7 Ultrasound Sensor

An ultrasound sensor was developed during the latter part of this research, by Mr. Lovell. The implementation and sample results are available in Appendix E. The ultrasound sensor was configured to measure the distance and the angle between the two shoes.

Working Principle

Ultrasound waves are mechanical acoustic waves at frequencies higher than can be heard by human ears (humans can hear waves up to at most 20 kHz). Ultrasound sensors typically have a transmitter, which generates the waves, and a receiver, which is excited when waves are encountered. Ultrasound waves travel at the speed of sound, so distances can be determined by measuring the amount of time between the start of transmission and the first incidence of reception. The speed of sound in dry air, at 20° C, is 331 m/s; this number does vary with temperature and humidity [76].

Earlier tests using two closely spaced receivers to detect phase difference (as in monopulse sonar [83]) provided poor results, hence a large separation between receivers was used with a time-difference-of-arrival method.

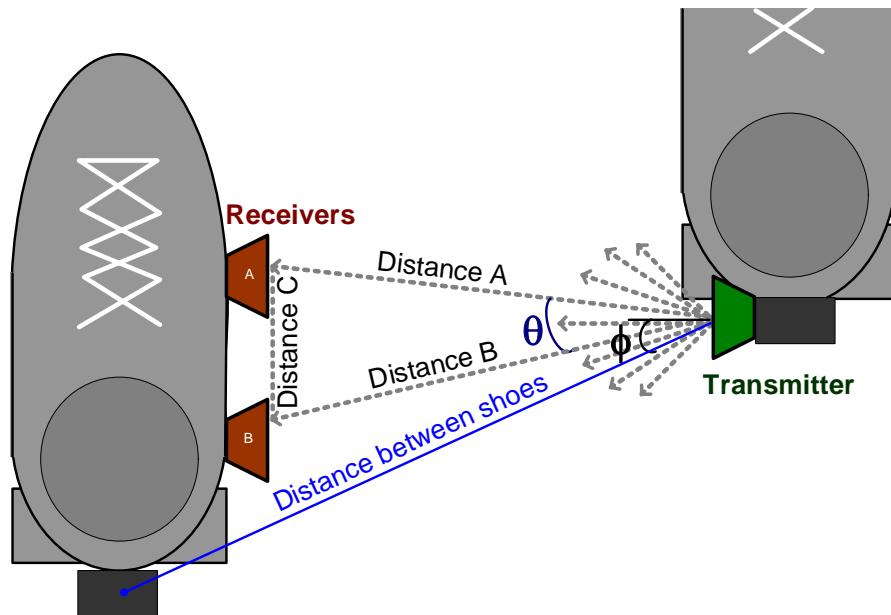


Figure 3.35 Ultrasound sensor for distance and angle between two shoes

To measure the distance and relative position between the shoes using the time-difference-of-arrival, a transmitter was placed on the right shoe, and two receivers were placed on the left shoe, as shown in Figure 3.35. On the right shoe, the transmitter, shown in green, generated ultrasound waves at 40 kHz. On the left shoe, the ultrasound electronics measured the time which elapsed before each of the receivers, shown in red, were first hit by the ultrasound waves. These time measurements led to an estimate of the distances from the transmitter to each of the receivers (A and B), and since the distance between the two receivers (C) is known, θ can be determined from the law of cosines, as described in Eq. 3.5. If desired, the distance from the transceiver to an arbitrary point (at known position in relation to A or B) and the corresponding angle from the horizontal, ϕ , can be determined from the geometrical relationship between the points.

$$\cos\theta = \frac{A^2 + B^2 - C^2}{2 \cdot AB} \quad (3.5)$$

3.4 Additional Components

The additional hardware and circuits used in this system are described in this section. Detailed part numbers and ordering information for all parts are listed in Appendix D.2.

3.4.1 Microcontroller

The microcontroller selected for the GaitShoe was the C8051F206, manufactured by Cygnal Integrated Products, Inc. The features of this 48 pin chip include its small size (8 mm x 8 mm, 1.20 mm thick), its on-board analog-to-digital convertor (ADC), and its low operating current (under 9 mA, typical). The analog-to-digital convertor features 12-bit resolution multiplexed in 32 channels, and has a net throughput of up to 100 kilosamples/sec (ksps). The chip was clocked at 22.118 MHz, and therefore has just over 22 million-instructions/sec (mips); the supply voltage provided was +3.3 V.

Figure 3.36 shows the implementation of the Cygnal C8051F206, as used in the GaitShoe system, and Figure 3.37 shows the placement of the components on the Main board. A six

pin JTAG programming header provided the ability to easily update the code on the chip as needed. Pin 14 must be connected to ground when the Cygnal chip is first turned on, in order to reset the ADC, so a manual switch was provided for this purpose. Pins 11 and 31 were connected to the 3.3 V power supply, each with 0.1 μ F capacitors to minimize noise. Five jumper pads ($RJ_1 - RJ_5$) were provided for future use with multiple stack boards; these can be used to give each of 32 boards a unique identification number.

A voltage reference was included on the board, shown connected to pin 7. This can be used to provide the C8051F206 with a reference voltage which is separate from the main power line; however, the use of this part was not implemented in testing.

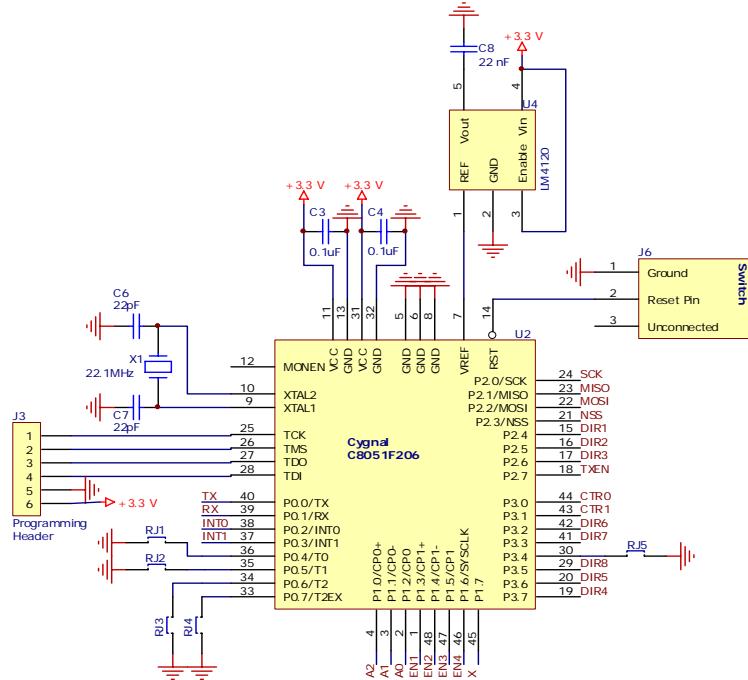


Figure 3.36 Schematic of the Cygnal C8051F206

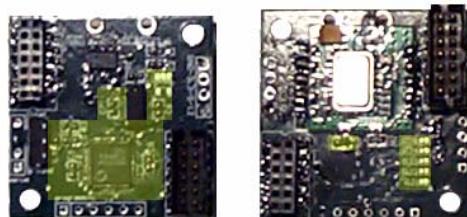


Figure 3.37 Photo of the main board, with the Cygnal C8051F206 components highlighted

3.4.2 Radio Frequency Transceiver

A critical requirement of the design of the GaitShoe was that it be an untethered system. Since the ability to use the GaitShoe for real time feedback was of strong interest, a wireless transceiver was required for data transmission.

Transceiver Hardware and Electronics

The wireless transceiver selected for use in this application was the DR3000-1 module, manufactured by RF Monolithics, Inc. The DR3000-1 module contains RF Monolithics' TR1000 amplifier-sequenced hybrid (ASH) transceiver on a daughter board, and was designed specifically for short-range wireless applications in the radio frequency (RF) range. The daughter board helps isolate the local ground from any noise on the main ground line, and contains most of the capacitors, resistors and inductors used to operate the TR1000. The TR1000 operates at 916.5 MHz, and transmits data at 115.2 kilobits/sec¹ (kbps), using amplitude-shift keyed (ASK) modulation. Surface-acoustic-wave (SAW) filtering is employed by the receiver (in two stages) to reject out-of-band noise and by the transmitter to suppress² output harmonics [85].

Figure 3.38 shows the schematic used for the DR3000-1, and Figure 3.37 shows the placement of the components on the Main board. Signals CTR0, CTR1, and TXEN were set by the microcontroller, to set the state (transmit or receive) of the transceiver. The microcontroller also sent signal TX, which contained the data to be transmitted. Signal RX was the data received by the transceiver, and was sent to the microcontroller. The NAND gate was implemented to ensure that pin 5 on the DR3000-1 was held low when the transceiver was operating in receive mode. It was necessary to add a 22 μ F capacitor across pins 7 and 8

-
1. Transmission at 115.2 kbps was selected so that the data could be converted to serial data, and transmitted to the computer over the serial line.
 2. SAW filters use the piezoelectric effect; a transducer at the input of the device converts electric waves into surface acoustic waves. The acoustic waves then excite a half-wavelength resonant acoustic cavity within the transducer, and this energy is coupled to a second resonant acoustic cavity (also of dimension corresponding half-wavelength) transducer at the output, where the signal is converted back to electric waves. Various methods of coupling are employed; SAW filters are commonly used in the telecommunications industry to suppress output harmonics [84].

(Vcc and GND) in order to minimize noise from the power line [85]; this increased the range of the system by from approximately 3 m to more than 15 m, during use in the MIT Media Lab.

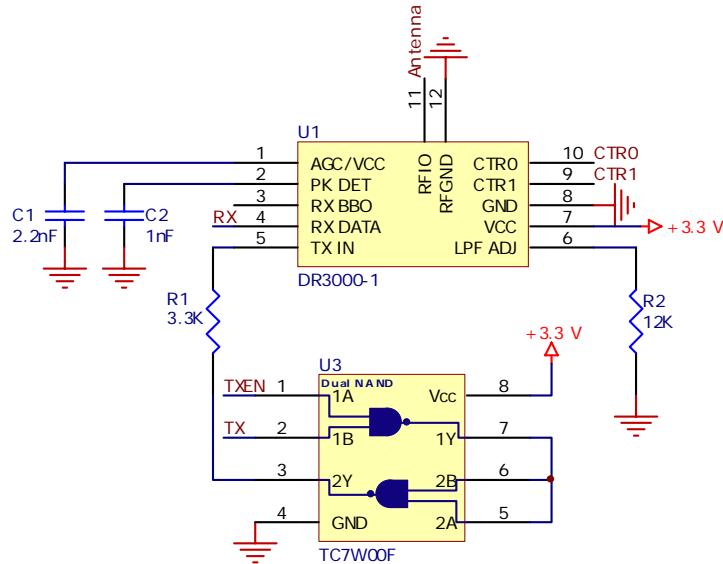


Figure 3.38 Schematic of the RF Monolithics DR3000-1

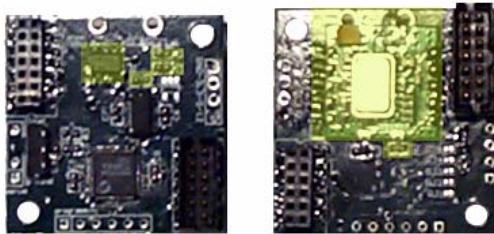


Figure 3.39 Photo of the Main board, with the components of the RF Monolithics DR3000-1 highlighted

The antenna used for most of the testing of the system¹ was a helical whip 1/4-wave antenna manufactured by Linx Technologies, Inc.; it has a female RP-SMA connector, and is omni-directional. It was connected to the DR3000-1 daughter board via a connector manufactured by Linx Technologies, which has a male RP-SMA connector with an 8.5"

1. A solid core wire cut to 8.2 cm in length (the 1/4 wavelength for 916.5 MHz) was used with the first three subjects. However, the wire was continually hit by the equipment used by the MGH Biomotion Laboratory, which caused it to bend back and forth and eventually shear off. The helical whip antenna was found to work nearly as well as the wire, and was more stable physically. Though it is recommended that it be used with a proximity groundplane, a groundplane was not implemented in this version of the system; doing so could improve the reliability of the wireless transmission.

length of coaxial cable attached with heat shrink and strain relief. The voltage standing wave ratio (VSWR) of this antenna is typically under 1.9.

RF Issues

RF Monolithics estimates that in the 916.5 MHz band, with transmission at 115.2 kbps, the operating distance will be up to 65.3 m in free space or as low as 8.1 m in dense cubical office space. Additionally, it is important to note that UHF radiation is well absorbed by the human body, particularly above 750 MHz, which will always be a confounding factor in optimizing such a wireless system.

The testing of the GaitShoe was carried out at the MGH Biomotion Laboratory (BML), which contained a large quantity of electrical equipment. In addition, the equipment of other residents in the building was unknown (for instance, some wireless phones operate in the 900 MHz band), and temporary magnetic resonance imaging (MRI) and positron emission topography (PET) laboratories are often set up outside the building. These factors were likely contributors to the low transmission range achieved during testing: the maximum range of the GaitShoe, while in use at the BML, was on the order of 3-5 m from the basestation. In contrast, in use at the MIT Media Lab, the maximum range was on the order of 15 m or more. However, the basestation was placed in the middle of the room at the BML which provided sufficient length for collecting data during the entire gait trial.

The 12-bit output from the ADC was converted to two balanced (equal numbers of zeros and ones) bytes, using the algorithm described in Appendix F.1. Error checking algorithms, such as cyclic redundancy check (CRC), were not implemented, but are recommended for future work with the GaitShoe.

3.4.3 Power

Several different voltages were needed in the GaitShoe system. While most of the hardware used +3.3 V, the Analog Devices gyroscope (ADXRS150) required +5 V, the Motorola electric field imaging device (MC33794DH) required +12 V, and the ultrasound sensor

required +9 V. To meet these requirements, and to allow the GaitShoe system to be untethered, a power board was developed, containing a replaceable +9 V battery and voltage regulators. The voltage regulators were three different fixed voltage output versions of the Analog Devices ADP1111 switching regulator. For fixed outputs of +3.3 V and +5 V, the ADP1111 operated in step-down mode, and for the fixed output of +12 V, the ADP1111 operated in step-up mode. In steady-state operation, the hardware on each GaitShoe attachment drew a current just under 50 mA, while the basestation drew just under 15 mA.

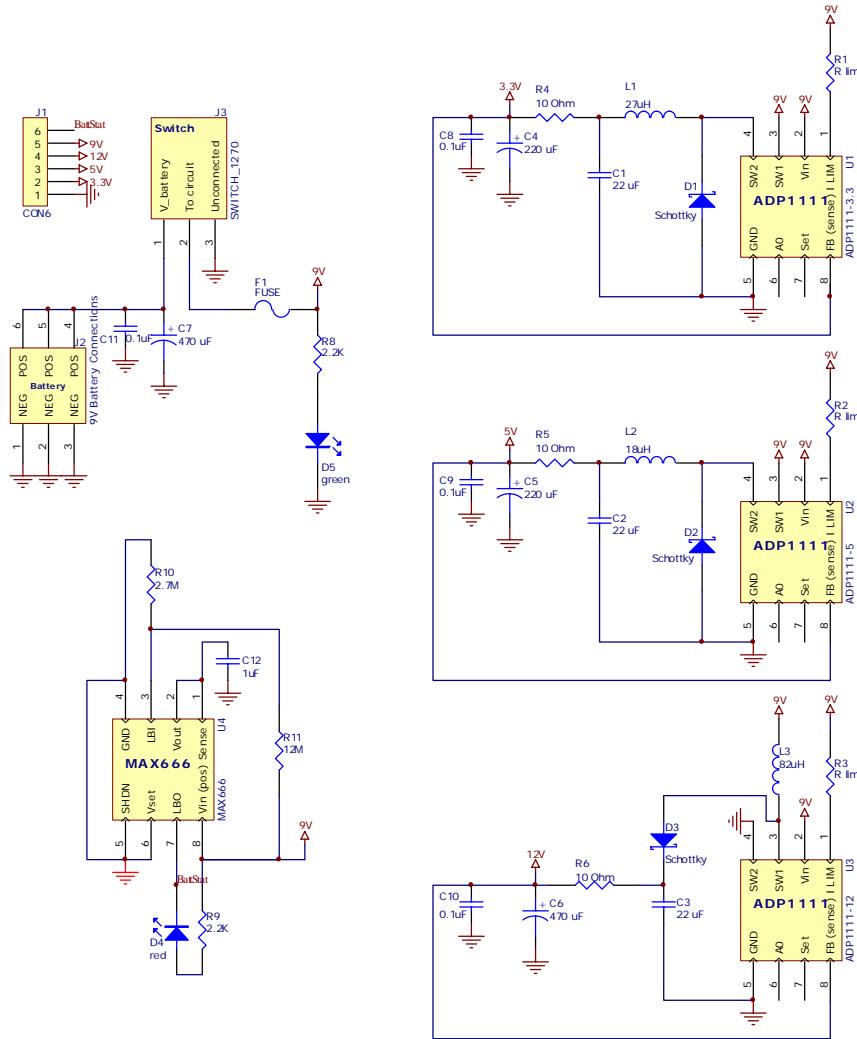


Figure 3.40 Schematic of the Power Board

The schematic showing the circuits on the power board is shown in Figure 3.40. As implemented, R_1 , R_2 , and R_3 , were each set to 0 k Ω (the resistor pads were included to provide

the option to limit the current; this was unnecessary). The inductor values were selected based on allowing the battery to reach a minimum voltage of 7V. Inductors are available in a finite range of values, so the closest lower value was selected. This resulted in used (and calculated) values of: $L_1 = 27 \mu\text{H}$ (30 μH), $L_2 = 18 \mu\text{H}$ (19 μH), and $L_3 = 82 \mu\text{H}$ (98 μH). The direct battery output was used to provide the +9 V for the ultrasound sensor. To minimize noise, each voltage line had a ceramic 0.1 μF capacitor, C_8 , C_9 , C_{10} , and C_{11} , as well as an electrolytic capacitor, C_4 , C_5 , C_6 , and C_7 ; the electrolytic capacitors were set to 220 μF on the +3.3 V and +5 V lines and 470 μF on the +12 V and +9 V lines.

The power board also contained a MAX666 chip, which was used to detect when the battery level becomes low. The MAX666 pulls pin 7 high when the battery level is above the threshold set by resistors R_{10} and R_{11} and low when the battery level falls below the threshold. With $R_{10} = 2.7 \text{ M}\Omega$ and $R_{11} = 12 \text{ M}\Omega$, the threshold was set at 7 V. The level of pin 7 was provided as an output, and was connected to the +9 V battery source via a light emitting diode (LED), L_2 , such that the LED was lit when the battery level fell below the threshold of 7 V. This provided a visual indication that the battery level was low. A second LED, L_1 , was located between the +9 V battery source and ground, which provided a visual indication that the power board is turned on.

Finally, the power board also contained a resettable fuse, in series with the on-off switch, and before the rest of the hardware. Although a +9 V battery cannot source enough current for a shock to be felt through the skin, a fire or burn hazard from a short circuit remains a possibility, hence, a fuse was included as a safety precaution. The fuse used was part MF-SM030, a positive-temperature coefficient (PTC) polymer fuse manufactured by Bourns, Inc. PTC polymers change from a low resistance to a very high resistance state in response to a spike in current above the trip current level; PTC polymer fuses are resettable by power cycling the device (and removing the cause of the high current). The MF-SM030 has a trip current of 0.6 mA.

3.4.4 Other Components

An analog multiplexer from Analog Devices, part ADG608, was used for the collection of data from multiple sensors. The ADG608 switches between each of eight inputs to a common output. Its features include a high switching speed (typical transition time to switch inputs is 120 nsec), low power dissipation (1.5 μ W, max), and a low on resistance (30 Ω , max).

All op-amps used in the system were the Analog Devices OP262. The OP262 is a dual op-amp, which has a low offset voltage (50 μ V, typical), low supply current (.5 mA, typical), high slew rate (10 V/ μ sec, typical) and low noise (0.5 μ V peak-to-peak, typical). Nearly all were operated at +3.3 V, except for the op-amps used in the ultrasound sensor circuits, which were operated at +9 V.

In the basestation, the RS-232 driver used to convert the streaming wireless data to serial data was a MAX233 chip, manufactured by Maxim. This chip runs on +5 V, and provides the higher RS-232 voltage with a charge pump, hence it does not require any external components.

A number of connectors were used in the GaitShoe system. The most critical selection was the choice of connectors between the stacked printed circuit boards. These provided both electrical interconnections, and some mechanical stability. The 26 total interconnections were split between the two headers (14 for one, 12 for the other), which were located at opposite corners. The connectors were Molex Milli-Grid shrouded headers and mating receptacles, and were rated for 100 insertion cycles (reasonable for prototyping)¹. The net names on the interconnections are shown in Figure 3.41.

1. The footprint of these connectors and the foot print of the RF Monolithics DR-3000-1 board (the largest component) determined the basic size of the board: 4.5 cm x 3.7 cm, the smallest size which could accommodate these three components. A smaller single connector with 26 pins wa originally used, which allowed for a smaller board footprint, however, this connector was not robust enough when mounted on the back of a shoe.

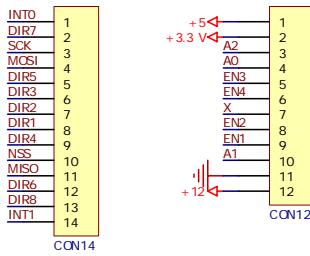


Figure 3.41 Header interconnection net names

3.5 The GaitShoe System

The design of the GaitShoe has resulted in a wearable wireless system which can measure a vast number of parameters about gait. A photo of the GaitShoe hardware on a pair of shoes is shown in Figure 3.42, and the structure of the GaitShoe system is summarized in a high-level block diagram in Figure 3.44. Figure 3.44 shows the (uncalibrated) outputs across all the sensors used during the subject testing.



Figure 3.42 GaitShoe hardware

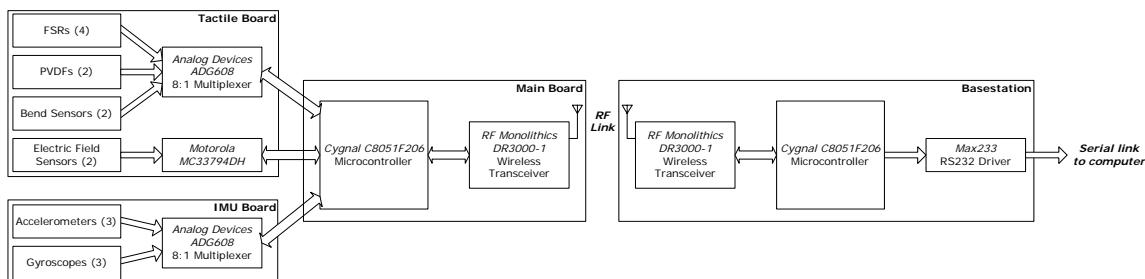


Figure 3.43 High level block diagram of the GaitShoe system

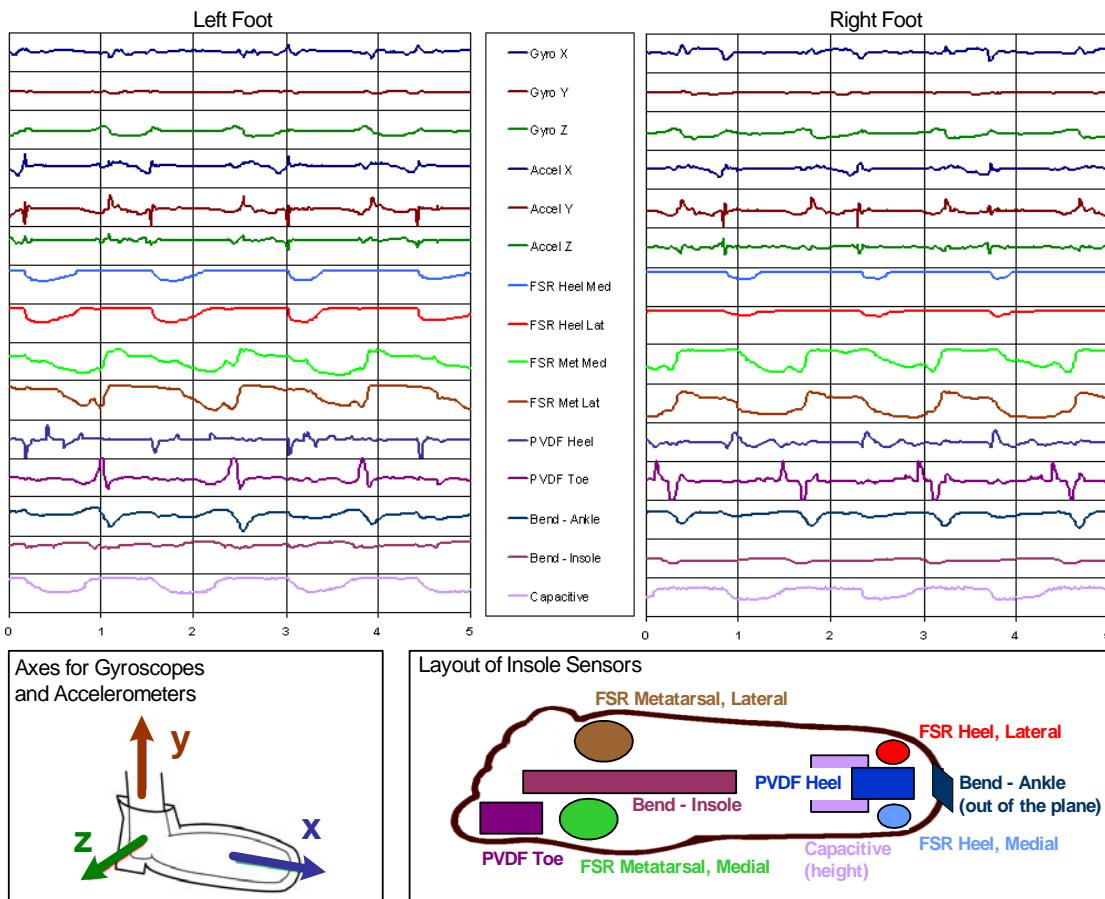


Figure 3.44 Sensor outputs across all sensors for both feet

Chapter 4

SENSOR ANALYSIS

This chapter discusses the pre-processing applied to the GaitShoe data, the analysis techniques and assumptions used, and presents an analysis of the output of all the sensors, including calibration of the most relevant sensors. The outputs of the calibrated sensors in the GaitShoe system were analyzed to generate clinically relevant gait parameters; the validation of these results by comparison to data collected simultaneously by the system at the Massachusetts General Hospital Biomotion Lab is discussed in Chapter 5.

Calibration of the sensors was necessary because sensor outputs may vary, depending upon, *inter alia*, manufacturing tolerances of sensors and other components in the circuits with the sensors. Although manufacturers provide specifications ("specs") for sensors, in general they are not sufficiently accurate. The sensitivity, which relates the output of the sensor to the standard units, must be determined for any sensor for which a quantitative output is desired. In addition, sensors such as the gyroscopes and accelerometers measure parameters that have positive and negative polarities. However, the actual sensors all have an output voltage greater than zero, so the zero offset for each of these sensors must be determined¹. The uncalibrated sensor outputs have been normalized to a scale of 0 to 1 by the maximum 12-bit analog-to-digital converter (ADC) value (i.e. the uncalibrated outputs were divided by 4095).

1. In a sensor which has a linear output, the zero offset corresponds to the y-intercept, and the sensitivity corresponds to the slope of the line.

4.1 Data Processing

Before the parameters derived in this chapter were used to calibrate and analyze the sensor outputs, a number of processing steps were applied to the data.

4.1.1 Truncation

The GaitShoe collects data continuously, while the MGH Biomotion Lab (BML) collected data for seven seconds during subject testing for validation of the GaitShoe. The GaitShoe data were truncated so that it started a couple seconds before the subject started walking, and concluded just before the subject stopped or turned at the end of the room. To determine the truncation¹, the raw data were loaded, and the outputs from several sensors on both feet were plotted, as shown in Figure 4.1.

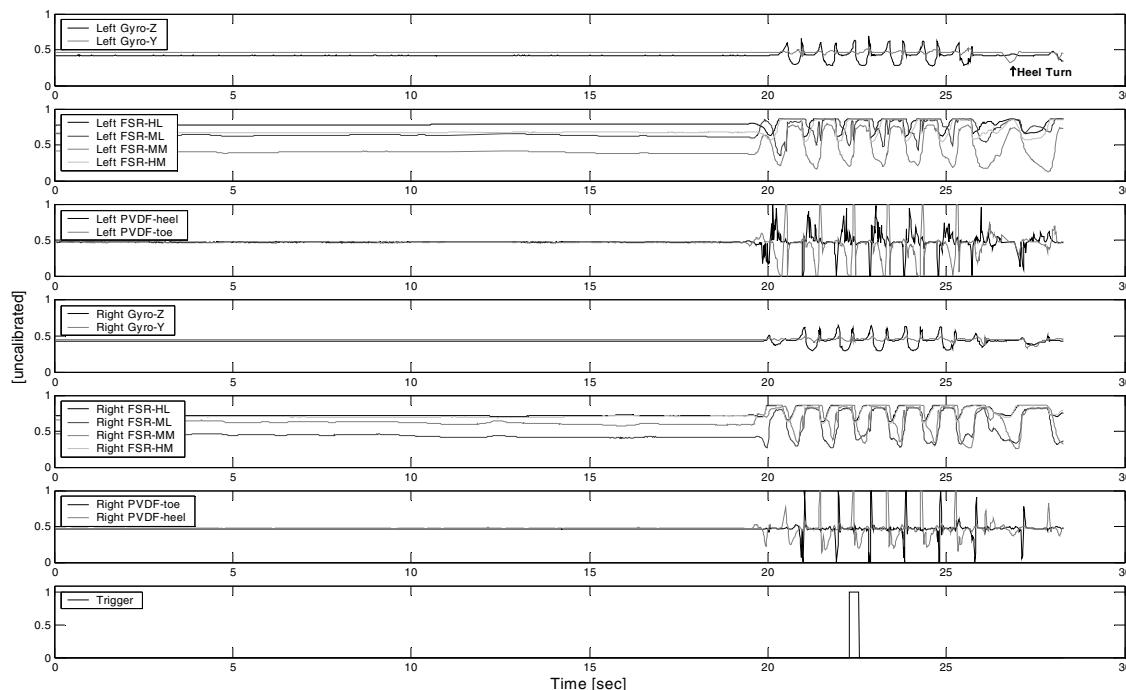


Figure 4.1 Complete raw output

The starting and stopping points were selected by investigation of plotted data. The initiation of gait was easily determined, because the subjects were instructed to stand still with

1. See timetrunc.m, an m-file available in Appendix F.2.

both feet on the floor before starting the gait trial; the starting point was selected approximately 2 seconds before the first changes seen in the sensors.

The stopping point was more difficult to determine. Subjects were told to stop when they reached a point a few feet away from the wall, and could either pause or continue walking by turning to their left. Because of this variation, the stopping point was set either when the subject slowed or turned, whichever occurred first. All four FSR outputs and both PVDF outputs, for both feet, were plotted for visualization of a slowing gait. For instance, in Figure 4.1, the left FSR output shows a broadened output during the last two steps, corresponding to a slowing gait. Two gyroscope outputs were plotted for both feet: the z-gyro, which measures the angular velocity corresponding to changes in the pitch of the foot, and the y-gyro, which measures the angular velocity corresponding to changes in the yaw of the foot. Turning on the heel of the foot results in changes picked up by the y-gyro only, this can be seen well in the left foot output in Figure 4.1 (this event is marked in the top graph of Figure 4.1). The stopping point was set just before either of these events occurred, and was always set mid-stance (load on the FSRs) for one foot, and just after toe-off (no-load on the FSRs) of the other foot.

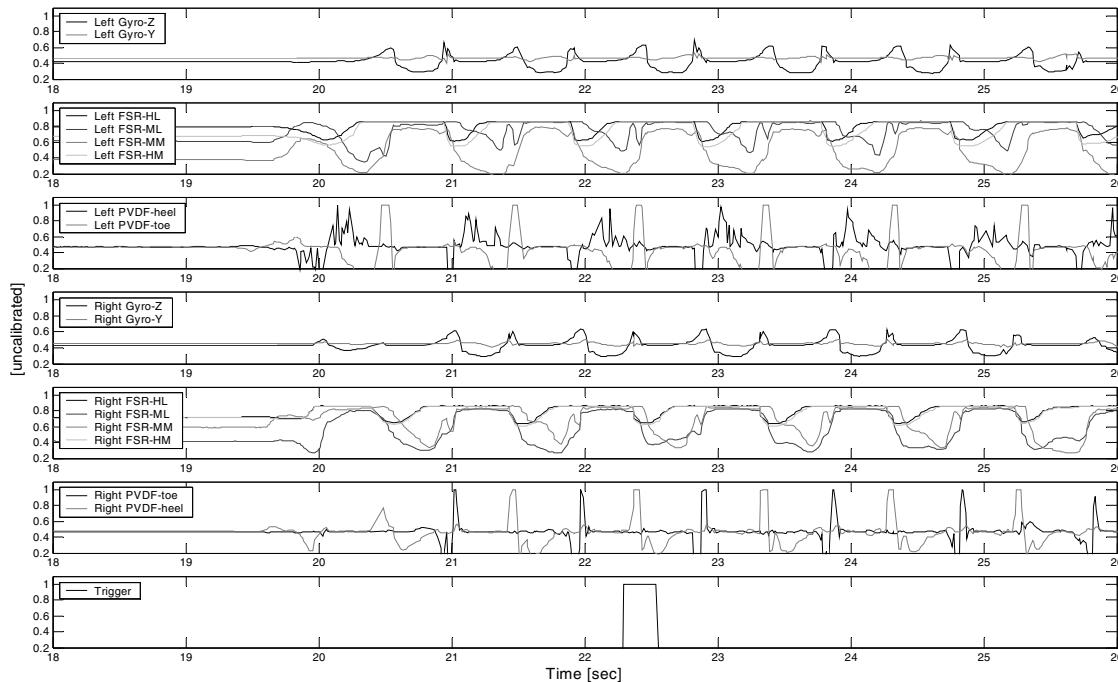


Figure 4.2 Truncated section of the raw output

The resulting truncated data are shown in Figure 4.2. All of the truncated gait trials for each subject were stored as a Matlab data file. The bottom graphs in both Figure 4.1 and Figure 4.2 is the output of the BML transistor-transistor logic (TTL) trigger, which was used to align the time scale of the GaitShoe with the time scale of the BML data (described in Appendix D). The initiation of the trigger flag corresponds to time zero on the BML system.

4.1.2 Time Adjustment

Stored with the data were two columns of information about the time points corresponding to each row of sensor outputs: the time as provided by the laptop computer clock, "comptime," and a timestamp provided by the clock and the microcontroller in the basestation, "basetime." The comptime stream was continuous, but was only accurate only to about 50 msec. The basetime stream was accurate to 0.1 msec, but it reset every 0.4096 seconds. The comptime was used to make initial observations of the data, such as for truncation, without needing to convert the basetime. It was also used as a moderately accurate reference during the basetime conversion; this reference was needed if the wireless connection failed and packets of data were not received by the basestation. The basetime was converted¹ to a continuous time stream using the comptime as a reference. The trigger output, shown on the bottom graph in Figure 4.2, was used to set time zero, corresponding to the BML time scale.

4.1.3 Data Adjustment

The final step² before calibration was to adjust the data³, by reordering the sensor columns, and by checking all data for outliers.

1. See timeadjuster.m, an m-file available in Appendix F.2.

2. See getshoedataorder.m, an m-file available in Appendix F.2, which ran timeadjuster.m and dataadjuster.m

3. See dataadjuster.m, an m-file available in Appendix F.2.

TABLE 4.1 Final sensor order

Column	Sensor Name	Abbreviation
1	FSR, under the medial heel	FSR-HM
2	FSR, under the lateral heel	FSR-HL
3	FSR, under the medial (first) metatarsal	FSR-MM
4	FSR, under the lateral (fifth) metatarsal	FSR-ML
5	PVDF, under the heel	PVDF-H
6	PVDF, under the great toe	PVDF-T
7	Bend, in the insole	Bend-I
8	Bend, at the ankle	Bend-A
9	Gyroscope, measuring about the z-axis	Gyro-Z
10	Accelerometer, measuring along the z-axis	Accel-Z
11	Gyroscope, measuring about the y-axis	Gyro-Y
12	Accelerometer, measuring along the y-axis	Accel-Y
13	Gyroscope, measuring about the x-axis	Gyro-X
14	Accelerometer, measuring along the x-axis	Accel-X
15	Electric Field Sensor, under the heel	EF-H
16	reserved for future use	
17	reserved for future use	
18	reserved for future use	
19	reserved for future use	
20	reserved for future use	
21	Time	
22	Outlier information	

Sensor Order

The data were reordered such that the columns of the data corresponded to the order of sensor outputs as listed in Table 4.1 (rows corresponded to time points). This was done mainly to set a standard order of sensor data across all subjects, since different versions of microcontroller code transmitted the sensor data in different orders. It was also used to group similar sensors together. The five columns reserved for future use were left empty for both the planned additional sensors, such as an electric field sensor under the metatar-

sals, and the ultrasound sensors, as well as any future sensors of interest (if more than five additional sensors were added to the system, additional columns could, of course, be added as well).

Outliers

As described in Section 3.4.2, the wireless transmission involved encoding the twelve bit sensor reading as two balanced bytes. The basestation received these data and used the serial line to send the computer the two balanced bytes, where they were decoded to reconstruct the twelve bit sensor reading. However, if a single bit (or 3, 5, or 7 bits) was not received correctly at the basestation, the two bytes could not be decoded to the original twelve bit number. If the decoding failed, the sensor output at that time point was flagged¹.

In addition, if two (or 4, 5, or 8) bits were not received correctly at the basestation, the resulting byte was still balanced, so it was able to be decoded and reconstructed as a twelve bit number; however, this reconstructed value was incorrect. Errors of this type resulted in outlying data points², which were identified using the first difference of the sensor data. For each point, the standard deviation of the first differences over a total of 75 points (which corresponds to about a minute's worth of data) were calculated. When evaluated, most points were located in the middle of the 75 points, but at either end of the data, a block of the first 75 or last 75 points were used.

A point was identified as an outlier if both of the following criteria were met: 1) the magnitudes of the first differences between the point and each of its neighbors were greater than the standard deviation times a threshold (the value of the threshold is discussed below), and, 2) the signs of those two first differences were opposite. The first criteria indicated that the value of the point changed rapidly with respect to its neighbors, and the

1. The computer program converted the twelve-bit output (range, 0-4095) to a normalized output with a range from 0-1; data which could not be decoded was set equal to 2.
2. See `findoutliers.m`, an m-file available in Appendix F.2.

second criteria indicated that the point was not along a valid trajectory of rapidly increasing (or decreasing) points. Each sensor data vector was evaluated twice, since outliers with a very large first difference could skew the mean standard deviation within the 75 point window for outliers which occurred earlier.

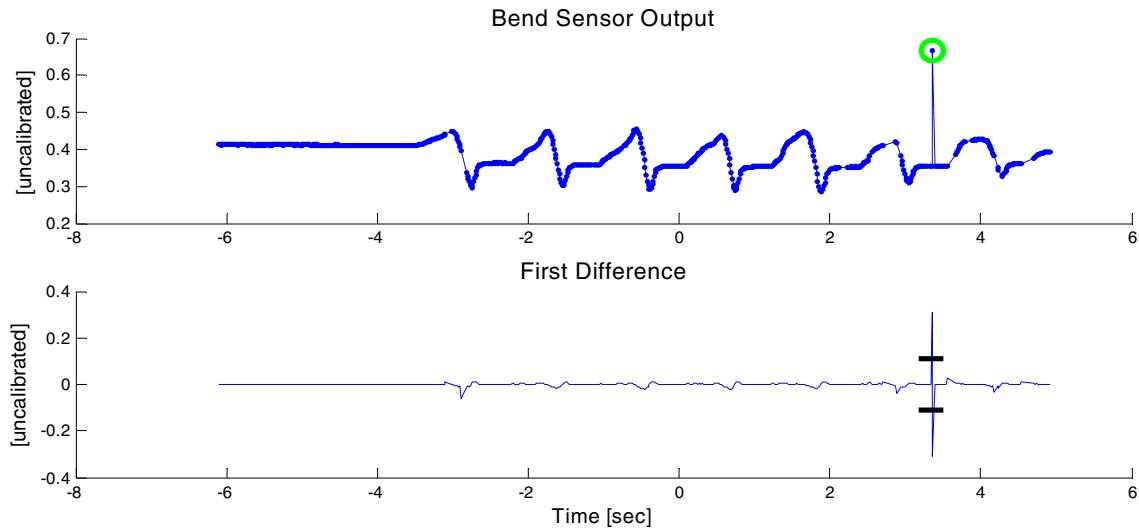


Figure 4.3 Outlier identified in bend sensor output

The FSRs, bend sensors, and electric field sensors all measured parameters which did not change rapidly, so a threshold of three standard deviations was used. Data from a bend sensor is shown in Figure 4.3, with the first differences plotted in the lower portion of the graph. One of the data points visually stands out as an outlier. The circle around this data point in the upper graph indicates that it met the criteria for being labeled an outlier. The horizontal lines on the bottom graph correspond to the value of the three standard deviations of the local first difference.

The accelerometers, gyroscopes, and PVDF strips were all high-bandwidth devices and measured parameters that changed very quickly, so a higher threshold of five standard deviations was used. Though many of the data points in the accelerometer output change very rapidly, only one met the criteria to be labeled outlier, as can be seen from the horizontal lines on the lower graph in Figure 4.4.

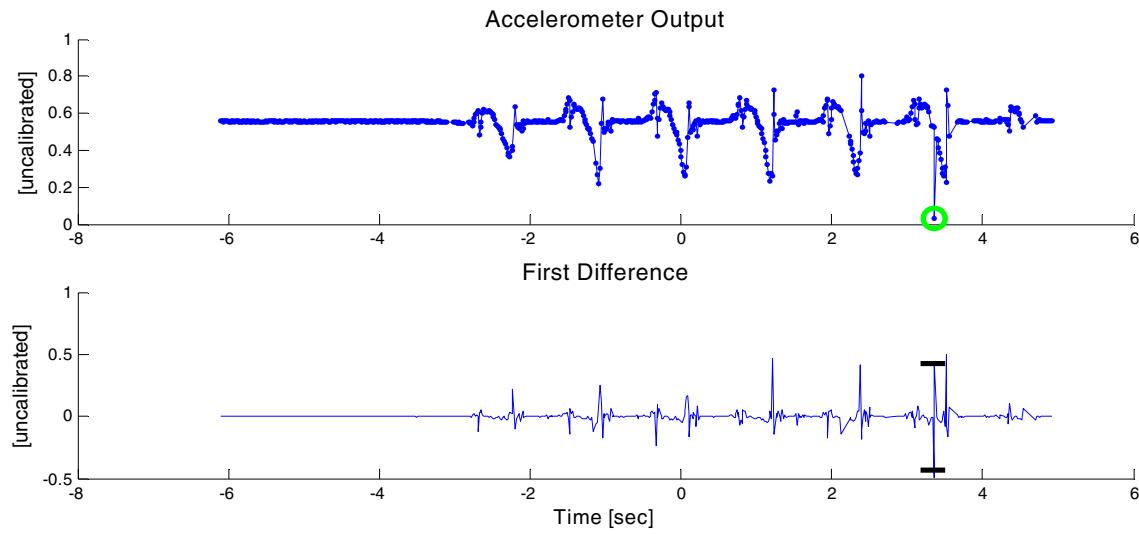


Figure 4.4 Outlier identified in accelerometer output

When a sensor data point was identified as either an odd-bit error from the flag, or as an even-bit error from the outlier evaluation, it was replaced by fitting a line¹ between the two neighboring points. Column 22 was used to store which data points were replaced, by using a base-18 number, where the position in the base-18 number corresponding to the column number was set to 1 if the sensor data point was determined to be an outlier, and set to 2 if the sensor data point was flagged as unable to be decoded (or left as zero if the data was unaltered).

Another wireless transmission problem was that some data packets were not received by the basestation. Each shoe was instructed to update every 0.0134 seconds, which corresponds to a data transmission rate of approximately 75 Hz. However, due to issues with the wireless transmission, as discussed in Section 3.4.2, there were occasional "dropped packets," where the time between data packets for a single shoe was more than 0.0134 seconds. An additional step² was performed on all of the data, where for any series of three dropped packets or fewer, data were generated for the missing time points by fit-

1. Polyfit.m, a standard Matlab function, was used with the polynomial degree set to 1 to fit the line, and polyval.m, also a standard Matlab function, was used to calculate the value at the time of the outlying sensor data point.

2. See gapfiller.m, an m-file available in Appendix F.2.

ting a line in the same manner as was used for the outliers. In this case, the corresponding row had a value of 0.1 placed in column 22.

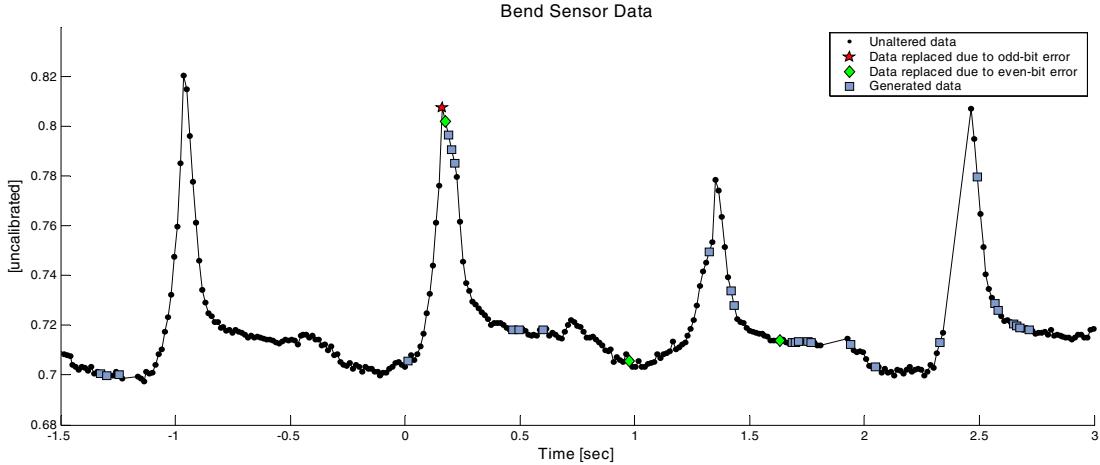


Figure 4.5 Bend sensor data with a number of adjustments

By using column 22 to store information about data replacement and data generation, it was easy to recover the status of every data point. A sample of bend sensor data which had all three types of data adjustments is shown¹ in Figure 4.5. Each resulting matrix of data, for the left and right feet over a single gait trial were stored in a Matlab data files for each subject.

4.1.4 Calibration and Analysis

To calibrate² the data, the truncated and adjusted data were loaded into Matlab. All rows with a non-zero value in column 22 were removed, because of concerns about the accuracy of the line-fit (the use of a spline-fit for data is discussed later in this chapter; this was done on an as-needed basis).

The accelerometers, gyroscopes, and bend sensors were calibrated using the sensitivities and zero offsets, and the FSRs were calibrated using the sensitivity functions; the determination of these parameters will be discussed in this chapter. The PVDFs were scaled to a

1. See `plotadjdata.m`, an m-file available in Appendix F.2.

2. See `gencalibrations.m`, an m-file available in Appendix F.2, was used to apply the calibrations to the data

centered around zero. The electric field sensor was only used on a few of the last subject, and so the electric field sensor output was not included in the calibrated data matrix; the data order is listed in Table 4.2. In addition, the sum of the four calibrated FSR outputs was calculated, and stored in column 20. The resulting calibrated data were stored in a Matlab data file.

TABLE 4.2 Order of calibrated and analyzed data

Column	Data	Abbreviation
1	Calibrated FSR, under the medial heel	FSR-HM
2	Calibrated FSR, under the lateral heel	FSR-HL
3	Calibrated FSR, under the medial (first) metatarsal	FSR-MM
4	Calibrated FSR, under the lateral (fifth) metatarsal	FSR-ML
5	Centered PVDF, under the heel	PVDF-H
6	Centered PVDF, under the great toe	PVDF-T
7	Calibrated Bend, in the insole	Bend-I
8	Calibrated Bend, at the ankle	Bend-A
9	Calibrated Gyroscope, measuring about the z-axis	Gyro-Z
10	Calibrated Accelerometer, measuring along the z-axis	Accel-Z
11	Calibrated Gyroscope, measuring about the y-axis	Gyro-Y
12	Calibrated Accelerometer, measuring along the y-axis	Accel-Y
13	Calibrated Gyroscope, measuring about the x-axis	Gyro-X
14	Calibrated Accelerometer, measuring along the x-axis	Accel-X
15	Pitch, determined from Gyro-Z	
16	Velocity in X _{Room}	
17	Displacement in X _{Room}	
18	Velocity in Y _{Room}	
19	Displacement in Y _{Room}	
20	Sum of calibrated FSR outputs (columns 1-4)	FSRsum
21	Time	

As will be discussed in this chapter, the z-gyroscope was analyzed to determine the foot pitch, and the pitch was stored in column 15. The x-accelerometer and y-accelerometer were used to determine the velocity and displacement along the x- and y- axes of the

room; the x-velocity and x-displacement were stored in columns 16 and 17, and the x-velocity and x-displacement were stored in columns 18 and 19. The final order of the calibrated and analyzed data is detailed in Table 4.2.

4.2 Analysis Model

4.2.1 Coordinate Systems

Analysis of the data requires the understanding of two coordinate systems, shown in Figure 4.6. The first coordinate system corresponds to the global reference frame of the room. The second corresponds to the local body frame, where the sensors are located and collect their measurements. Determination of room-based parameters such as orientation or position (for instance, corresponding to stride length) requires a transformation to the global reference frame.

For very simple motions, this transformation is straightforward. For example, a simple rotation about a single gyroscope axis is transformed to an angle of orientation within the fixed reference frame by integrating the corresponding gyroscope signal. Similarly a simple translation consisting only of motion along a single acceleration axis is transformed to a displacement within the fixed reference frame by double integration of the corresponding accelerometer signal. Of course, most motions of interest will consist of rotations about multiple axes, vector translations, and the two axes will be at some arbitrary position and orientation with respect to each other, requiring more complex transformations to obtain measurements of interest in the fixed reference frame of the room.

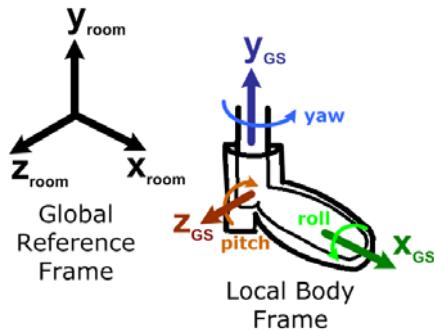


Figure 4.6 Frames of reference used in evaluation

4.2.2 Inertial Measurement Evaluation

The GaitShoe system contains three gyroscopes and three accelerometers, which form a strapdown "inertial measurement unit" [86], by collecting information about the three-dimensional motion of the foot through space. These measurements can be integrated with respect to time and combined to determine the position and orientation of the foot.

Evaluation of Complex Rotations and Translations

As discussed above, complex rotations and translations require careful evaluation to transform the motion in the local body frame back to the fixed reference frame. In particular, because orientation space wraps onto itself, and rotations do not commute, complex rotations in 3D space cannot be determined by examining only the rotations about each axis independently. For instance, in a given reference frame, rotation about the y-axis of $+30^\circ$ results in the same orientation as a rotation about the y-axis of -330° ; in addition, successive rotations about the x-axis and z-axis can result in the same final orientation. The two most commonly used methods of evaluation to deal with this issue are *Euler angles* and *quaternions*. *Euler angles* represent complex rotations as the product of three successive rotations about three orthogonal axes. *Quaternions* represent rotations via a three element vector, Q_v and a real number Q_r , where the magnitude of the rotation is $2\arccos(Q_r)$, and the rotation is about the axis described by Q_v . *Quaternions* can be transformed to *Euler angles*, and vice versa.

The Kalman filter is an important mathematical tool used to combine *quaternions* or *Euler angles* with translations in order to obtain the position and orientation. The Kalman filter was developed to provide a method of estimating a process with feedback from measurements [87]. The basic Kalman filter has two types of equations: time update equations, which predict the state of the process prior to the next time point; and, measurement update equations, which use the feedback from measurements at the time point to correct the estimation of the state. The basic Kalman filter is useful for estimating the state of a linear process; for estimating the state of a three-dimensional non-linear motion, the

extended Kalman filter, which linearizes about the current mean and covariance, is used (this initial analysis of the data did not include the implementation of a Kalman filter, though one is recommended for future work) [88] [89] [90].

Integration Considerations

Applying integration to gyroscope and accelerometer outputs results in a long-term increase in error of the position and orientation, because noise errors accumulate during integration via a random walk process; the root mean square of this drift is proportional to the square root of the time of the integration [91]. This results in a second source of error for the accelerometers. Because the accelerometers measure both dynamic and static acceleration, the contribution of gravity must be correctly subtracted before integration to determine position due to dynamic acceleration. As discussed in Section 4.4.2, this contribution of gravity is calculated by integrating the gyroscopes to determine the orientation of the foot. Errors in the integration of the angular velocity therefore contribute to errors in the calculation of the gravity contribution, which is subtracted prior to integration of the accelerometer signal.

These effects can be minimized by periodic recalibration of position and orientation, or by integrating over short time intervals. Because the subjects in this study of the GaitShoe were all walking, the gait included a state when each foot was flat on the floor. This state occurs between heel strike and toe off, and can be confirmed with other sensors, such as the four FSRs in the insole. This was exploited to improve the integration of the gyroscope and accelerometer signals: it was used to set the bounds of integration over a single stride, as well as to recalibrate the orientation about the z-axis and x-axis (both 0° when the foot is flat on the floor), and the position in the y-axis (0 cm). Though the positions in the z-axis and x-axis and the orientation about the y-axis in the horizontal plane do not have an external reference for recalibration, resetting the integration reduces errors. Healthy gait typically has a step rate close to 120 steps per minute, or 60 strides per foot per minute. As shown in Figure 4.7, stance time is typically 60% of the gait cycle, which results in integration over about 0.40 seconds; this short integration time should help reduce the random

walk noise accumulated during integration. For persons with gait which does not include foot-flat, such as toe-walkers, a different method will be required for recalibration. For instance, the bend sensor in the insole could be used to determine the orientation of the gyroscope every time the toes were on the floor, as detected by the FSRs located underneath the metatarsals.

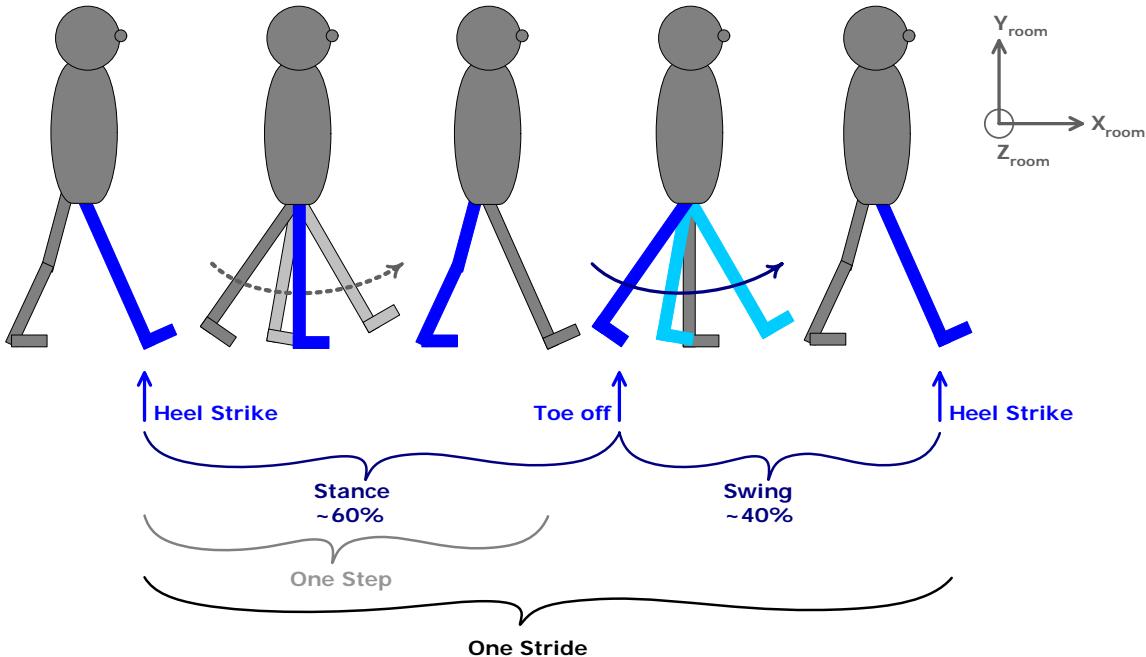


Figure 4.7 The gait cycle

Analysis Assumptions

The initial analysis of the GaitShoe sensor output was simplified by assuming that the data collected involved linear motion in the X_{GS} - Y_{GS} plane only (no translation in Z_{GS}), and rotation about the Z_{GS} axis only (the X_{GS} - Y_{GS} plane remained parallel to the X_{room} - Y_{room} plane). In other words, it was assumed that motion involved only changes portrayable on flat paper, such as in Figure 4.7.

The subject testing involved collecting data during walking, with the subject walking in a straight line only (as discussed in Section 4.1.1, the data were truncated before the subject turned upon reaching the end of the room). While this assumption is unlikely to be completely true for any of the subjects, it is a reasonable approximation to reality, particularly

for subjects with normal gait. Figure 4.8 shows the outputs of the three gyroscopes and three accelerometers from one of the gait trials during subject testing. Though there is acceleration in Z_{GS} , the magnitude is smaller than the accelerations measured in X_{GS} and Y_{GS} . In this sample, the standard deviation of the acceleration in Z_{GS} is 2.2 m/s^2 , compared to greater than 5 m/s^2 in X_{GS} and Y_{GS} , and the spread between the largest positive and largest negative acceleration in Z_{GS} is 20 m/s^2 , compared to greater than 45 m/s^2 in X_{GS} and Y_{GS} ; thus, the accelerations in X_{GS} and Y_{GS} are at least double those in Z_{GS} . Similarly, the angular velocity about Z_{GS} , is significantly larger in magnitude than the angular velocities measured about X_{GS} and Y_{GS} . In this sample, the standard deviation of the angular velocity about Z_{GS} is $164.5 \text{ }^\circ/\text{s}$, compared to less than $35 \text{ }^\circ/\text{s}$ about X_{GS} and Y_{GS} , and the spread between the largest positive and largest negative angular velocity about Z_{GS} is $781.2 \text{ }^\circ/\text{s}$, compared to less than $270 \text{ }^\circ/\text{s}$ in X_{GS} and Y_{GS} ; thus, the angular velocities about X_{GS} and Y_{GS} are less than a third of the angular velocity about Z_{GS} .

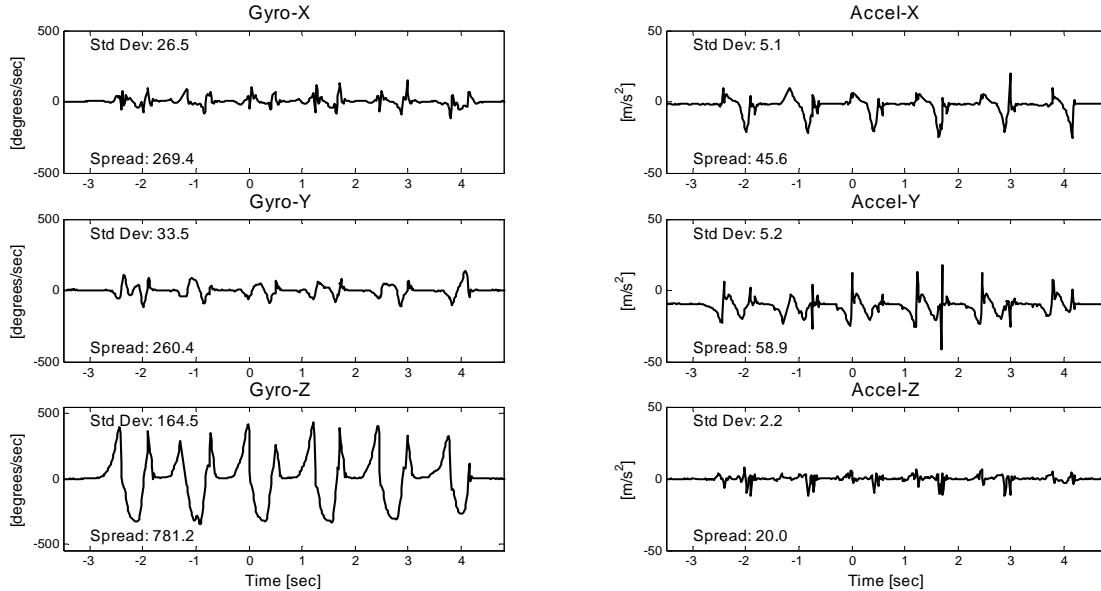


Figure 4.8 Comparison of IMU outputs during walking gait

For this initial analysis of the GaitShoe system, these assumptions were reasonable, without requiring the complex mathematics required to represent true 3D motion. Thus, the motion analysis consisted of integrating the output of the z-gyroscope to obtain the pitch of the foot about Z_{room} , and transformation of the single and double integration of the out-

put of the x- and y-accelerometers (after incorporating the pitch of the foot) to obtain the velocity and displacement along the X_{room} direction.

Methods of Integration

Two methods of integration were used in the analysis, a linear integrator, and Simpson's integration with a spline-fit. For the latter, a spline¹ was fit to a subset of the data and used to evaluate the integral numerically, using adaptive Simpson quadrature².

The linear integrator used the assumption that the data collection rate of $\Delta t = 0.0134$ sec between data updates was a small enough interval that the change in the gyroscope or accelerometer output could be considered linear between the two time points, as shown in Figure 4.9 on sample gyroscope output.

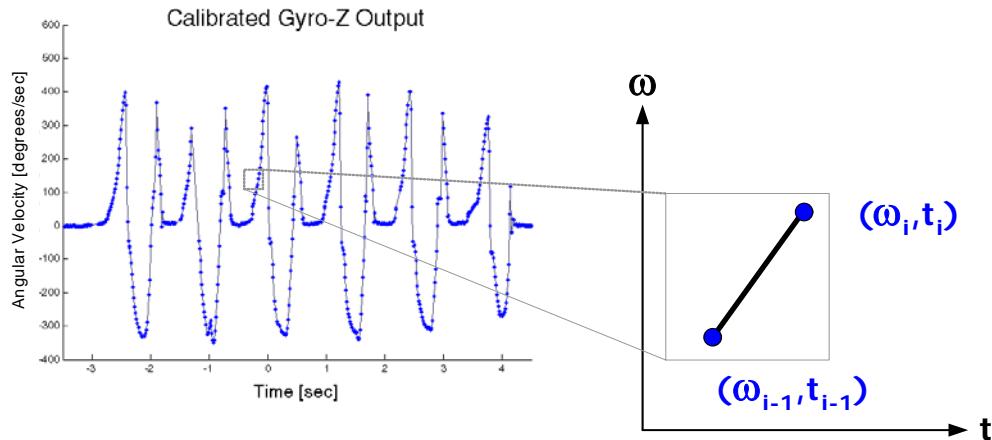


Figure 4.9 Illustration for linear integration

The equation of the line between two time points can be defined by its slope, m ,

$$m = \frac{(\omega_i - \omega_{i-1})}{(t_i - t_{i-1})}, \quad (4.1)$$

and its y-intercept, b ,

$$b = \omega_{i-1} - m \cdot t_{i-1}. \quad (4.2)$$

1. See spline.m, a standard matlab function was used for the spline-fit.

2. See quad.m, a standard matlab function was used to integrate using adaptive Simpson quadrature.

Then, the integral of the angular velocity between time point t_{i-1} and t_i is:

$$(\Theta_i - \Theta_{i-1}) = \int_{t_{i-1}}^{t_i} (m \cdot \tau + b) d\tau, \quad (4.3)$$

which reduces to

$$\Theta_i = \frac{1}{2} \cdot m \cdot (t_i^2 - t_{i-1}^2) + b \cdot (t_i - t_{i-1}) + \Theta_{i-1}. \quad (4.4)$$

Equations 4.1, 4.2, and 4.4 provided a simple method¹ of linearly integrating the data.

4.3 Gyroscopes

All three gyroscopes were calibrated; the calibrated z-gyroscope output was integrated to determine the pitch of the foot.

4.3.1 Calibration

For the gyroscopes, two types of information were required: the zero offset, to center the output around zero; and, the sensitivity, to convert the output to units of °/s.

Zero Offset

The zero offset of the gyroscopes was simply the output of the gyro when the hardware was at rest. The results are summarized in Table 4.3.

Sensitivity

The sensitivity of the gyroscopes was determined by rotating each gyroscope about its sensitive axis, through a range of constant angular velocities.

A precision gearhead motor from Globe Motors (#409A582) was used to rotate the gyroscopes. This motor is a high torque permanent magnet motor with a no-load rating of

1. See linear_integrator.m, an m-file available in Appendix F.2.

3000 rpm at +12 V. As purchased, the assembly is geared down such that the gearhead output shaft has a no-load rating of 25 rpm at +12 V, with 400 ounce/inches of torque; the motor functions from 6-12 V. During calibration, the polarity of the inputs were reversed in order to switch the direction of rotation, and the voltage input was varied to adjust the angular velocity. A flat metal turntable was attached to the gearhead output shaft, and the stack hardware was mounted directly to the metal turntable. The rotational velocity was measured with an optical tachometer.



Figure 4.10 Photo of the turntable used for the gyroscope calibration

For each gyroscope, the stack hardware was placed on the turntable shown in Figure 4.10, and taped securely into place, with the sensitive axis of the gyroscope undergoing calibration visually aligned over the center point of the turntable. Data were collected for 20 to 30 seconds each at several discrete angular velocities, and the clipped means¹ were calculated² for each set of data. A line was fit to the clipped mean data points and the zero offset value (which corresponds to 0°/s); the slope of the fit line is the sensitivity of the sensor. The results are summarized in Table 4.3, including the coefficient of correlation of the line fit, and the line fit for gyroscopes on IMU-1 is shown in Figure 4.11. Note that the Murata

1. The mean, calculated after discarding the top 10% and bottom 10% of the data.

2. As discussed in Section 4.1.3, the GaitShoe data may contain some incorrect data points, as a result of the wireless transmission protocol. For data collected about gait, these points are identified by looking at the first difference; here, any spurious points were eliminated by using the clipped mean.

gyroscope (axes Y and Z) specification indicated a sensitivity of 3.05×10^{-4} [output/($^{\circ}$ /s)] and a zero offset of 0.5 [output]. The Analog Devices gyroscope (axis X) specification indicated a typical sensitivity of 8.93×10^{-4} [output/($^{\circ}$ /s)], a maximum sensitivity of 9.82×10^{-4} [output/($^{\circ}$ /s)] and a zero offset of 0.5 [output].

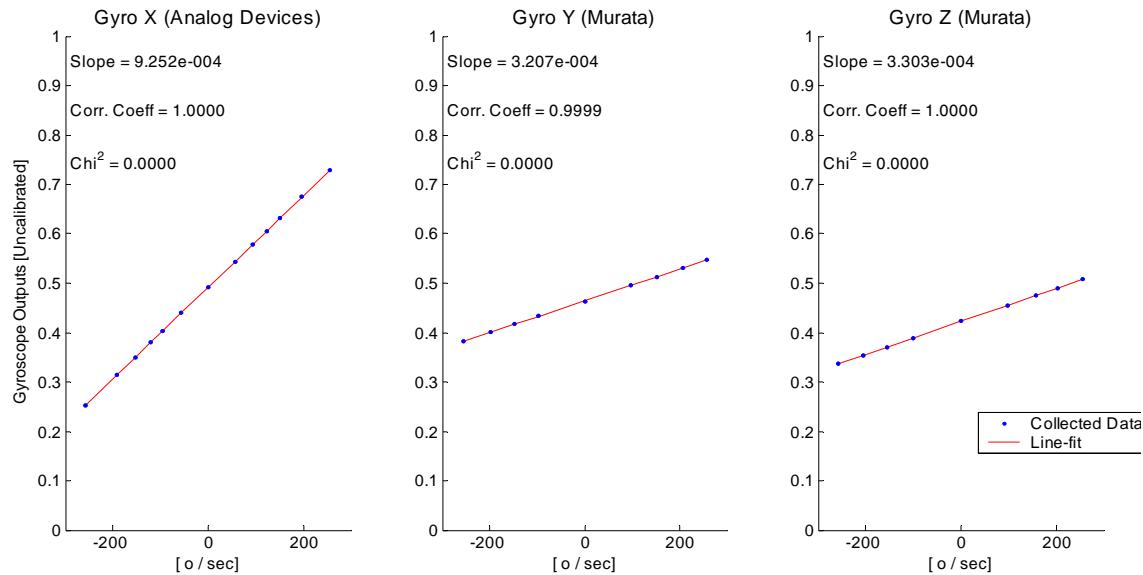


Figure 4.11 Line fits for determining the sensitivity of the gyroscopes (shown for IMU-1)

TABLE 4.3 Gyroscope sensitivities and zero offsets

Gyroscope	Sensitivity^a		Zero Offset
	Slope	Coeff. of Corr.	
IMU-1 X	9.25×10^{-4}	1.000	0.492
IMU-1 Y	3.21×10^{-4}	0.999	0.462
IMU-1 Z	3.30×10^{-4}	1.000	0.423
IMU-2 X	9.33×10^{-4}	1.000	0.508
IMU-2 Y	3.32×10^{-4}	1.000	0.446
IMU-2 Z	3.23×10^{-4}	1.000	0.432

a. Sensitivity units are [output/($^{\circ}$ /s)]; all other units are the uncalibrated output (scaled from 0-3.3 V to 0 to 1

4.3.2 Analysis of the Pitch

The pitch of the foot was determined by integrating the z-gyroscope output. The sign of the pitch follows the convention used at the MGH Biomotion Lab (BML), where in facing a subject such as the one pictured in Figure 4.7, a positive rotation about the z-axis corresponds to a rotation from the y-axis to the x-axis (left hand rule). The z-gyroscope output was integrated over a single stride. A variety of approaches for integration were developed.

The integration bounds for the gyroscope were determined from the times of heel strike and toe off. The midpoint between the heel strike time and the toe off time was calculated and was used for the starting and stopping point of each integration.

Direct Linear Integration

The first method¹ of determining the pitch used the calibrated z-gyroscope data and linear integration. A sample is shown in Figure 4.12, where the integration bounds are indicated with a red dot. The problem with this approach is that the integration sometimes fails to return the angle to 0° after a single stride. In Figure 4.12, the first integration is significantly off, and the second and fourth integrations are slightly off, while the third integration looks very reasonable.

There are two reasons which usually account for the failure to return the angle to 0° at the end of the stride: dropped packets; and drifting z-gyroscope output. In the first integration, there is a fairly large gap of dropped packets, which results in a miscalculation of the angle at the first time point following the gap, and a shift of the subsequent angles. This problem can be most readily addressed by improving the wireless implementation to prevent packet loss or by providing a method of on-board data storage. For this work a number of methods using spline-fits were developed to generate the missing data.

1. See `gyroz_linint.m`, a m-file available in Appendix F.2.

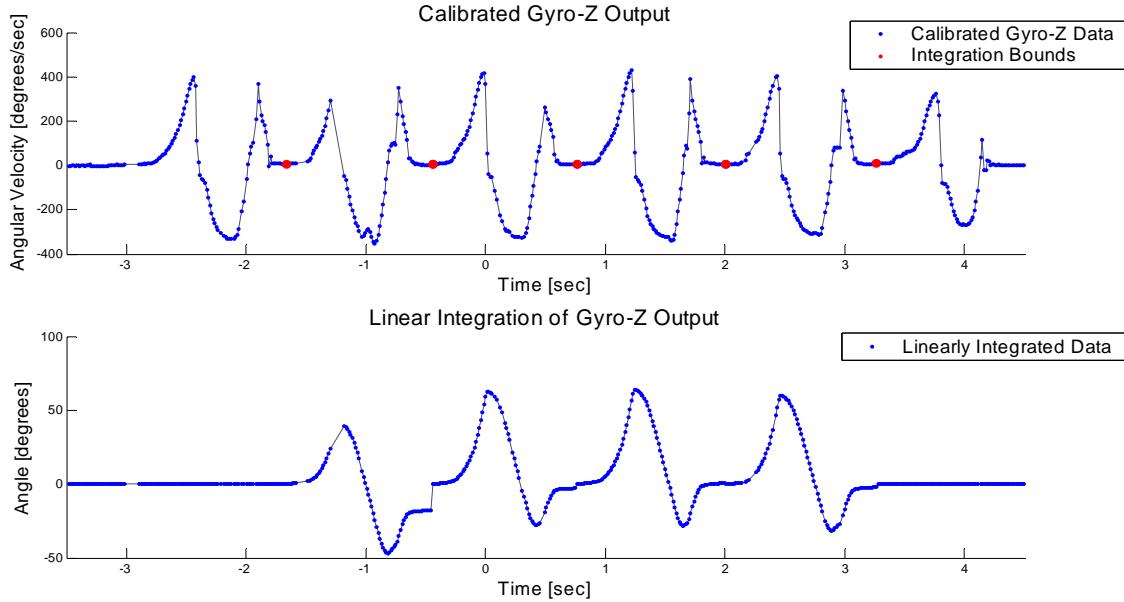


Figure 4.12 Sample of direct linear integration of the z-gyroscope

The z-gyroscope output corresponding to the fourth integration interval is shown in Figure 4.13. The output of the z-gyroscope is not zero during stance in this sample. This is because during calibration of the gyroscope, the zero-offset determined in Chapter 4 was used, but gyroscope output actually drifts very slightly over time. To compensate for this, an iterative method was developed.

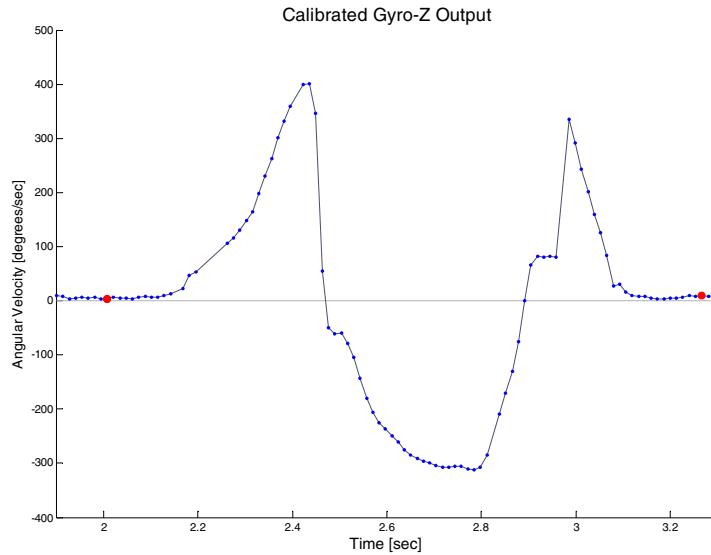


Figure 4.13 Z-gyroscope output where angular velocity is non-zero during stance

Iterated Linear Integration

To compensate for the slight drifting of the gyroscope's zero-offset, the second method¹ of determining the pitch linearly integrated the z-gyroscope data, but checked the final value of each integration between each set of integration bounds. An "endlimit" value was defined as 0.1% of the full-scale across all integrations. In Figure 4.12, the full-scale is approximately -50° to 60° , so the endlimit value would be 0.1% of 110° , or 0.11° .

For a given pair of integration bounds, if the final value had a magnitude greater than the endlimit, a small "nudge" value was added to the calibrated z-gyroscope data between these integration bounds. The magnitude of the nudge value was equal to the difference between the endlimit and the final value after integration, and the sign of the nudge value was opposite from the sign of the final value after integration. The linear integration between these integration bounds was repeated. This process was repeated until the final value was within the endlimit.

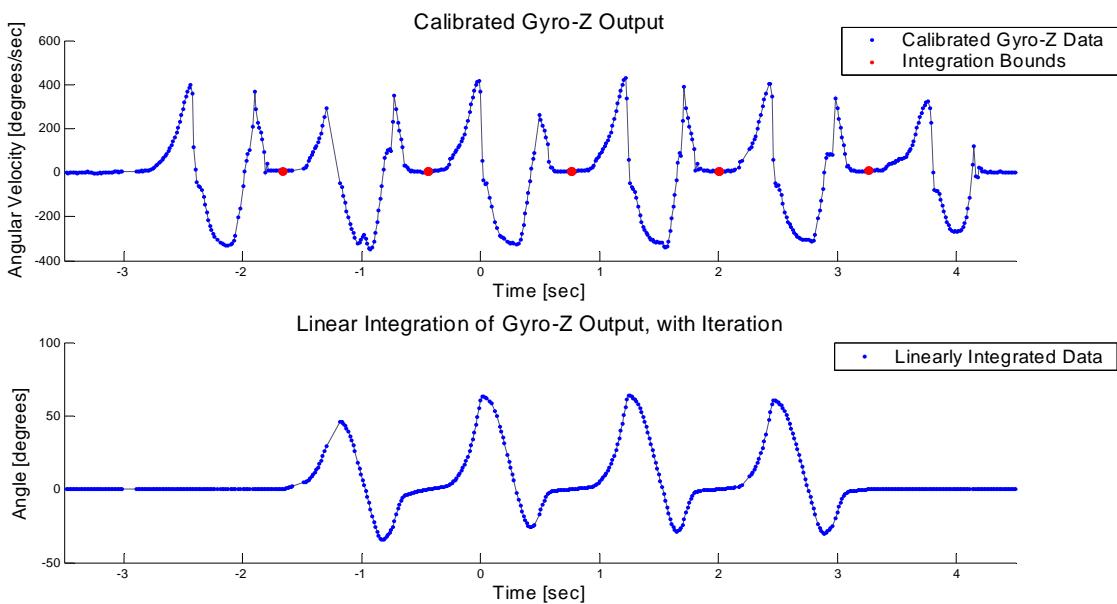


Figure 4.14 Sample of linear integration of the z-gyroscope, with iteration

1. See `linear_integrator_endadjust.m` and `gyrozlinint_withspline.m`, m-files available in Appendix F.2.

The results of this method are shown in Figure 4.14. All four of the integrations, including the first with dropped packets, now return smoothly to 0° .

Iterative Linear Integration with Spline-fit and Iterative Simpson's Integration

The third method¹ added an additional step to compensate for any gaps of dropped packets. After the iterative linear integration was complete, the data were checked between each set of integration bounds for gaps with more than four packets dropped in a row. If such gaps were found, a spline was fit to the calibrated z-gyroscope data. Then, the spline was used with Simpson's integration to determine the angle corresponding to the time points where packets were dropped (an iterative integration was used, similar to the one described above).

The results of this method are shown in Figure 4.15. There were three gaps of significant length, two during the first iteration, and one during the final iteration. The results of the spline-fit fill in the missing data points, but do not significantly affect the overall calculation of the angles.

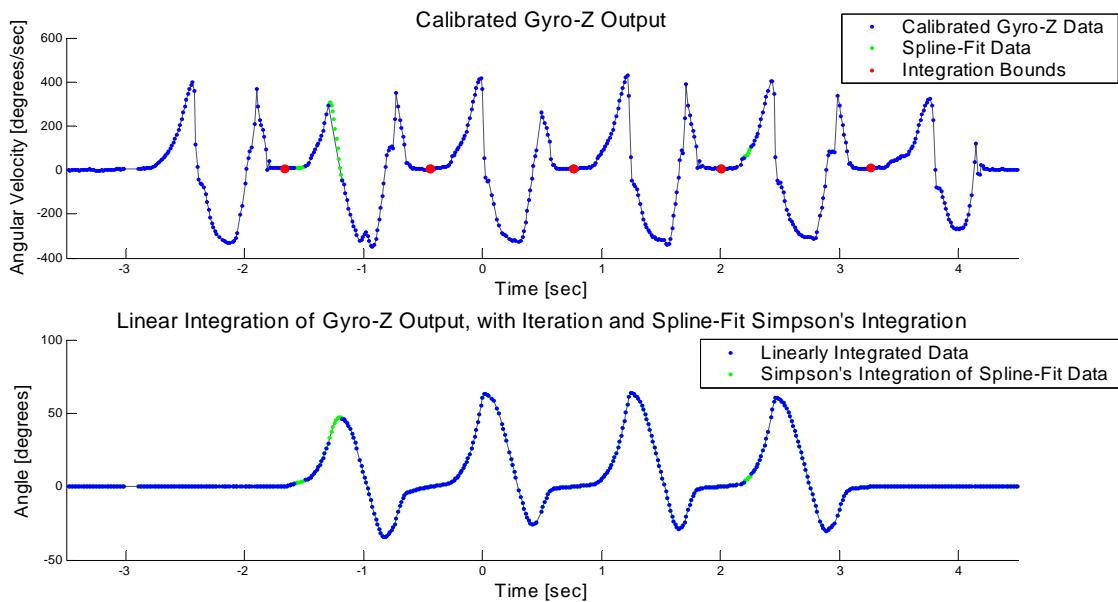


Figure 4.15 Sample of linear integration of the z-gyroscope, with iteration and spline-fit

1. See `simpsonsintegrator_endadjust.m` and `gyrozlinint_withspline.m`, m-files available in Appendix F.2.

Iterative Linear Integration with Post-Integration Spline-Fit

The final method¹ was developed as an alternative to the previous method; it was computationally expensive to do the spline-fit and iteration, particularly since the overall results were not changed. In this method, after the z-gyroscope was linearly integrated with iteration as described above. As necessary, any area of interest in the pitch was fit with a spline for evaluation.

For evaluating the performance of the pitch calculation with the BML results (see Chapter 5), the minimum and maximum values of the pitch, and the times at which they occurred, were compared. The maximum and minimum pitch during the gait cycle are labeled in Figure 4.16.

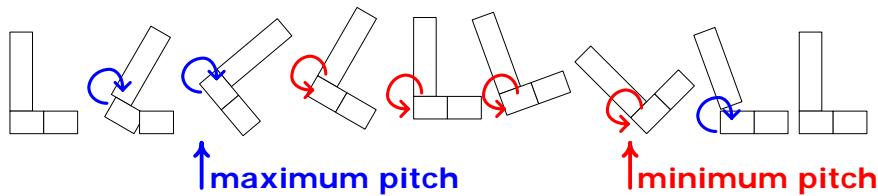


Figure 4.16 Pitch of the foot during the gait cycle

Each minimum and maximum of the calculated pitch were located, and a spline was fit to the pitch data over nine points (with the extremum at the center; this corresponds to 0.1 sec of data if no packets are dropped). The spline was fit with time points every 3.34 msec, which is four times as many data points as are collected if no packets are dropped. The results of this method are shown in Figure 4.17.

The maximum or minimum of each spline-fit (and the corresponding times) were determined, and compared to the maximum and minimums (and times) of the foot pitch information from the BML data. Figure 4.18 shows the BML data plotted with the integrated pitch. The shape of the pitch iteratively integrated from the angular velocity closely resembles the shape of the BML foot pitch

1. See `gyroz_postspline.m`, an m-file available in Appendix F.2.

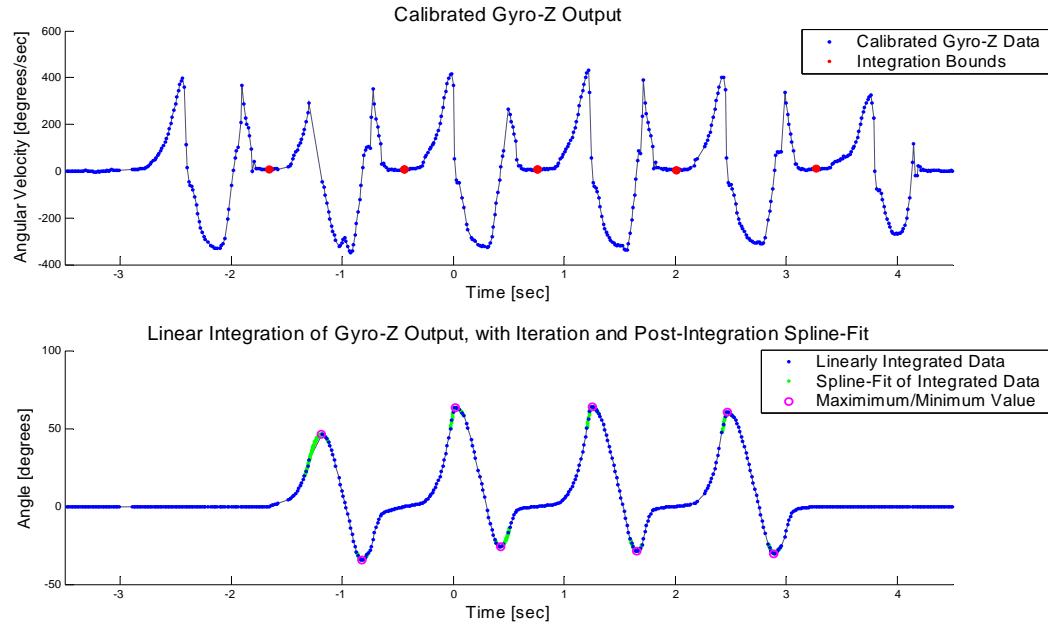


Figure 4.17 Sample of linear integration of the z-gyroscope, with iteration and post-integration spline-fit

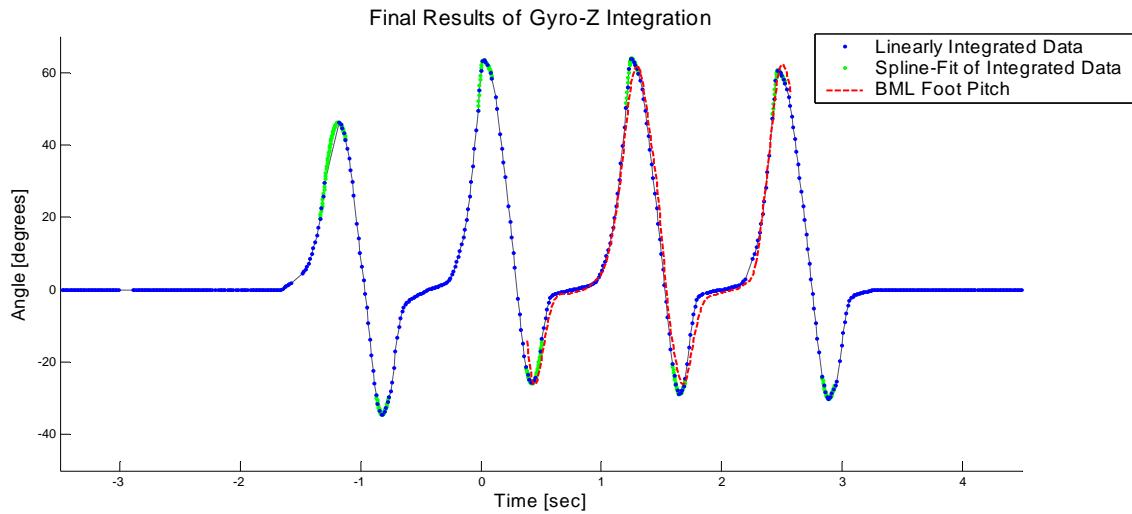


Figure 4.18 Final results, compared to BML data

The root mean square error (RMS error) between these two curves was calculated. As shown in Figure 4.19, a spline was fit to the linearly integrated data at timepoints corresponding to the timepoints of the BML data. The RMS error calculated between these two curves was 3.4° .

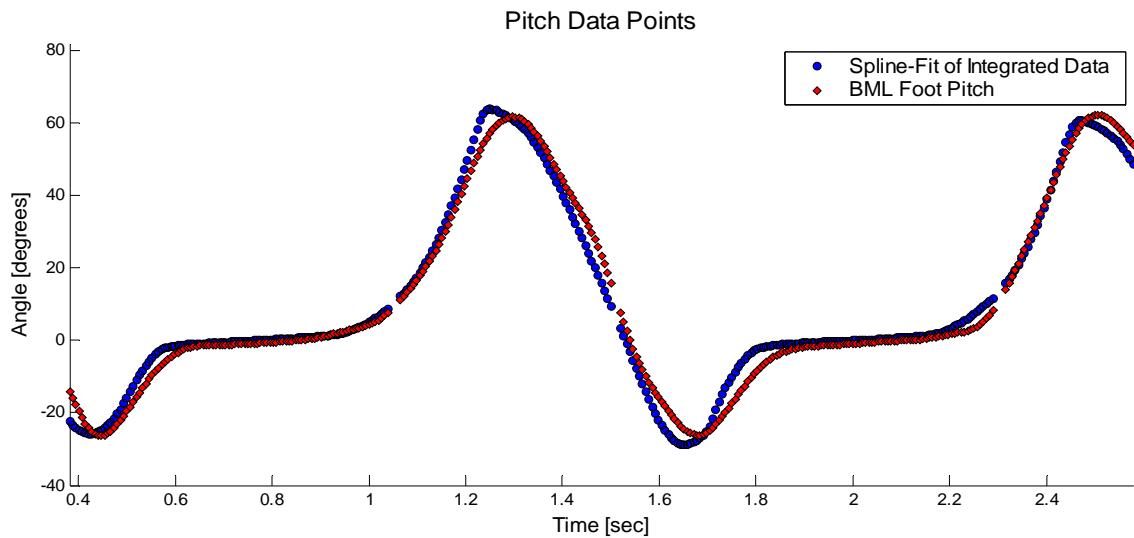


Figure 4.19 Comparison of BML pitch to GaitShoe pitch

4.4 Accelerometers

All three accelerometer outputs were calibrated; the calibrated x- and y- accelerometer outputs were used to determine the velocity and displacement of the foot in the X_{room} and Y_{room} coordinates.

4.4.1 Calibration

Three types of information about the accelerometers were required: the zero offset, to center the output around zero; the sensitivity, to convert the output to units of m/s^2 ; and the orientation of the accelerometers relative to the foot, to interpret the sensor output accurately.

Zero Offset and Sensitivity

Determining the sensitivity of the accelerometers was very straightforward, by using gravity. Naturally, the gravitational acceleration vector, \mathbf{g} , is stable, accurate, and readily available. By rotating the sensor such that the axis of interest was orthogonal to the earth's surface, and then rotating the sensor 180° , measurements of $+1 \text{ g}$ and -1 g were easily obtained. The sensitivity, $V_{\text{sensitivity}}$ (units: output/ m/s^2), of the acceleration measurement

is described by Eq. 4.5, where V_{1g} is the measurement at +1 g and V_{-1g} is the measurement at -1 g.

$$V_{sensitivity} = \frac{1}{2g} \cdot (V_{1g} - V_{-1g}) \quad (4.5)$$

The zero offset, $V_{zero-offset}$ (units: output), was set as the midpoint between the +1 g and the -1 g measurements, as described by Eq. 4.6.

$$V_{zero-offset} = \frac{1}{2} \cdot (V_{1g} + V_{-1g}) \quad (4.6)$$

The positioning of the accelerometers with respect to the gravity vector was done by hand. The hardware was slowly rotated about each of the three axes; this allowed each accelerometer to sweep through the gravitational acceleration vector, \mathbf{g} , twice. As shown in Figure 4.20, rotation about the x-axis resulted in the y-axis and z-axis accelerometers sweeping through \mathbf{g} ; similarly, rotation about the z-axis resulted in the x-axis and y-axis accelerometers sweeping through \mathbf{g} , and rotation about the y-axis resulted in the x-axis and z-axis accelerometers sweeping through \mathbf{g} .

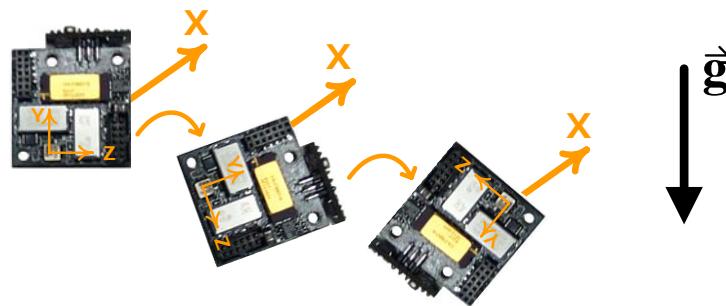


Figure 4.20 Demonstration of accelerometer calibration

Rotating by hand provided the ability to slightly modify the orientation of the accelerometers during rotation, in order to place the accelerometer axes parallel to \mathbf{g} , and maximize the sensor reading. The two axes within an accelerometer are specified to be perpendicular to each other to within 0.01° , and are parallel to the sides of the accelerometer to within 1° [71]. The accelerometer on the face of the IMU measures along the y-axis and the z-

axis, and the accelerometer located along the top of the board measures along the x-axis. Both accelerometers were soldered to the board by hand, so while the x-axis is expected to be close to perpendicular to the y-axis and the z-axis, the exact angle was unknown.

The data were low-pass filtered¹, with second-order Butterworth coefficients² and a low-pass cutoff frequency of 2 Hz, and the sampling frequency was 75 Hz. The rotation by hand was performed slowly, with the goal of keeping the rotation at a constant speed, as the goal was to measure only gravitational acceleration. The filtering was used to remove any small acceleration changes resulting from hand jitter. The values of V_{Ig} and V_{-Ig} for each calibration were determined by finding the maximum and minimum values of each output (the uncalibrated output of all sensors has been scaled from 0 to 1). A sample graph of the filtered accelerometer outputs is shown in Figure 4.21.

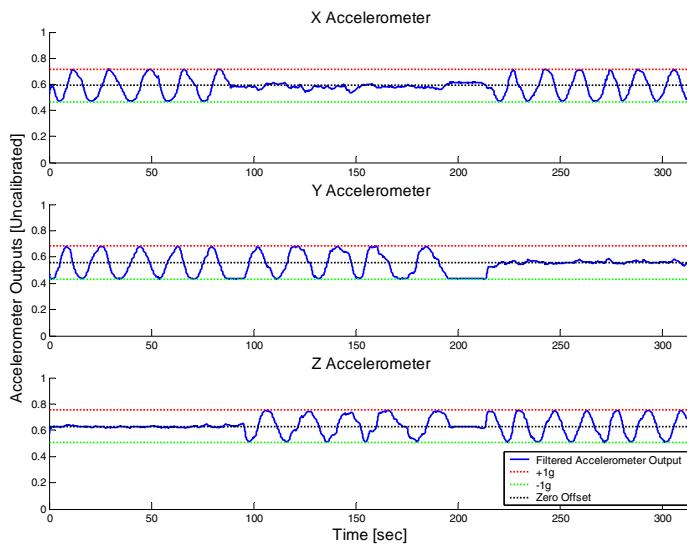


Figure 4.21 Sample accelerometer calibration for determining sensitivity and zero-offset.

The calibration was carried out seven times, on each IMU³, throughout the subject testing. The values of V_{Ig} and V_{-Ig} for each IMU were determined from the average of the results from all seven calibrations. The sensitivity and zero offset for each axis of the accelerom-

1. Matlab function `filtfilt.m`, a backwards and forwards filter, was used.
2. Matlab function `butter.m` was used to generate the Butterworth coefficients.
3. Two IMUs were used during testing of the GaitShoe system, one per foot; this document refers to the two IMUs as IMU-1 and IMU-2.

eters on each IMU were then determined from Eq. 4.5 and Eq. 4.6, using the mean values of V_{Ig} and V_{-Ig} ; results are summarized in Table 4.4, and shown in Figure 4.21. Note that the Analog Devices specification indicated a typical sensitivity of 0.11 [output/m/s²], and a maximum sensitivity of 0.14 [output/m/s²]; typical zero offset was indicated as 0.5 [output], with a maximum of 0.7 [output].

TABLE 4.4 Accelerometer sensitivities and zero offsets

Accelerometer	+1 g Output		-1 g Output		Sensitivity ^a	Zero Offset
	Mean	Std Dev	Mean	Std Dev		
IMU-1 X	0.676	0.001	0.425	0.001	0.13	0.551
IMU-1 Y	0.694	0.002	0.442	0.002	0.13	0.568
IMU-1 Z	0.619	0.005	0.375	0.002	0.12	0.497
IMU-2 X	0.716	0.002	0.467	0.002	0.13	0.591
IMU-2 Y	0.684	0.002	0.433	0.004	0.13	0.558
IMU-2 Z	0.750	0.003	0.507	0.003	0.12	0.628

a. Sensitivity units are [output/(m/s²)]; all other units are the uncalibrated output (scaled from 0-3.3 V to 0 to 1].

4.4.2 Orientation

Determining the orientation of each accelerometer in space was critical because the acceleration due to gravity must be correctly subtracted from the total acceleration signal during the analysis of gait data.

For simple rotation about the z-axis, the orientation of the foot can be determined by integrating the angular velocity about **z** to obtain the pitch of the foot. Then, the pitch orientation of the accelerometer is determined by subtracting the angle of inclination of the accelerometer with respect to the foot from the pitch, to allow the contribution of the gravity vector to be appropriately subtracted from the total acceleration measured.

The orientation of the accelerometer with respect to the foot is different for each subject, as the size and shape of shoe influence the inclination of the GaitShoe attachment on the shoe. When a subject stands still, with both feet flat on the floor (the foot flat on the floor corresponds to a pitch of 0°), each accelerometer measures only the fraction of gravitational acceleration along its sensitive axis. Therefore, as shown in Figure 4.22, the angle α_x can be determined as described by Eq. 4.7., where A_x is the acceleration measured by the x-accelerometer with the foot flat on the floor.

$$\alpha_x = \arcsin \frac{A_x}{g} \quad (4.7)$$

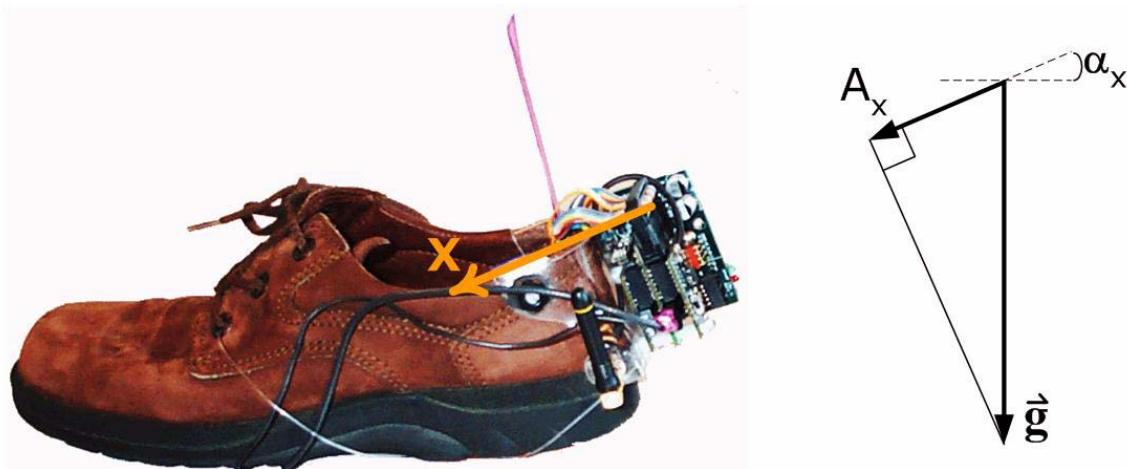


Figure 4.22 Determination of the orientation of the accelerometers

The angle α_y can be similarly determined; the difference between these two angles provides the relative orientation between the x- and y-accelerometers in the x-y plane.

4.4.3 Velocity and Stride Length Analysis

Velocity and stride length were determined by single and double-integration, respectively, of the acceleration along the X_{room} axis, using the output of the x- and y-accelerometers. In addition, the vertical velocity and vertical displacement were determined by single and double-integration, respectively, of the acceleration along the Y_{room} axis. The accelerome-

ters are fixed to the back of the shoe via the GaitShoe hardware, and their orientation with respect to the room and the gravity vector changes as the subject walks.

Determination of Dynamic Acceleration

The acceleration measured by the x-accelerometer can be resolved into two components, corresponding to the dynamic acceleration from foot motion, and the static acceleration due to gravity, $Ax_{dynamic-dyn}$, and Ax_{static} respectively. Figure 4.23 shows these components, and the relationship between the gravity contribution and the static component, given the pitch orientation of the x-accelerometer, Φ_x .

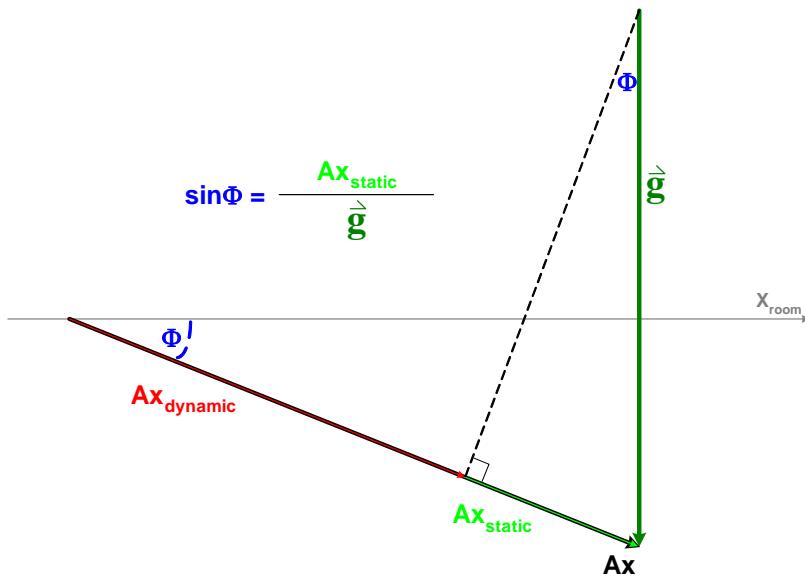


Figure 4.23 Dynamic acceleration along X_{room} only

In order to calculate the contribution of Ax_{static} component, the orientation of the x-accelerometer with respect to the room, Φ_x , must be determined. At time point t_i , $\Phi_x(t_i)$ is calculated from the sum of the pitch at the same time point, t_i , $\Theta_x(t_i)$, and the orientation of the x-accelerometer with respect to the horizontal, α_x :

$$\Phi_x(t_i) = \alpha_x + \Theta_x(t_i), \quad (4.8)$$

where α_x is calculated as described in Eq. 4.7 on page 126 (the sign of α_x has been determined by the right-hand rule, so Θ_x , which was determined using the left-hand rule to correspond to the BML data, is multiplied by -1).

The dynamic acceleration measured by the x-accelerometer at time point t_i , $A_{x\text{dynamic}}(t_i)$, is calculated by subtracting¹ the contribution of gravity from the total acceleration:

$$A_{x\text{dynamic}}(t_i) = A_x(t_i) - \bar{g} \cdot \sin \Phi_x(t_i). \quad (4.9)$$

The dynamic acceleration measured by the y-accelerometer is determined using the same method, but substituting the appropriate y-variables.

Determination of Appropriate Acceleration for Integration

The total acceleration vector experienced by the foot, A_{foot} , can be resolved into two components corresponding the reference frame of the room, $A_{x\text{-room}}\text{dynamic}$ and $A_{y\text{-room}}\text{dynamic}$. These both contribute the dynamic component measured by the x-accelerometer, $A_{x\text{dynamic}}$, as determined above, and are shown in Figure 4.24

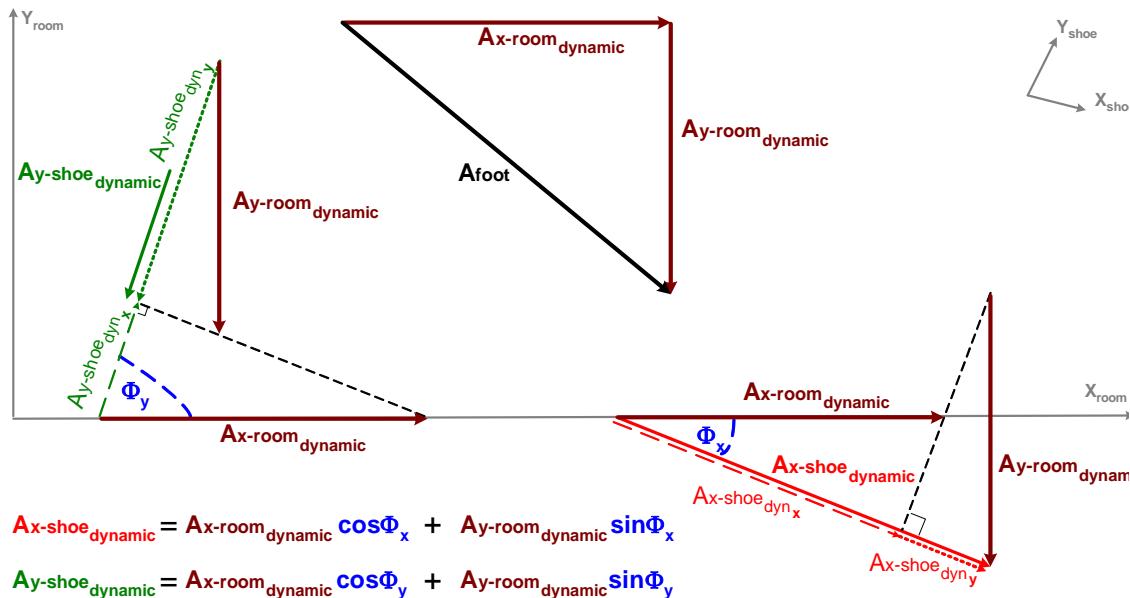


Figure 4.24 Dynamic acceleration with both X_{room} and Y_{room} components

1. Although Einstein's Equivalency Principle states that dynamic and static acceleration cannot be determined from measurement of acceleration alone, given the orientation of the object (e.g. Θ_x), the static component resulting from gravitation acceleration can be determined and subtracted from the total vector acceleration. However, any error in the orientation will contribute to the error in the dynamic component.

The most accurate way to determine the actual dynamic acceleration along X_{room} , which is the axis of interest for the velocity and the displacement calculations, is to combine the outputs of both the x-accelerometer and the y-accelerometer. The y-accelerometer has a dynamic component, $A_{y,dynamic}$, that also measures components of both $A_{x,room,dynamic}$ and $A_{y,room,dynamic}$, as shown in Figure 4.24. Using the output of both the x- and y-accelerometers, $A_{x,room,dynamic}$ and $A_{y,room,dynamic}$ can be determined from:

$$A_{x,room,dynamic} = \frac{A_{x,dynamic} \cdot \sin\Phi_y - A_{y,dynamic} \cdot \sin\Phi_x}{\cos\Phi_x \cdot \sin\Phi_y - \cos\Phi_y \cdot \sin\Phi_x}, \quad (4.10)$$

and

$$A_{y,room,dynamic} = \frac{A_{x,dynamic} \cdot \cos\Phi_y - A_{y,dynamic} \cdot \cos\Phi_x}{\sin\Phi_x \cdot \cos\Phi_y - \sin\Phi_y \cdot \cos\Phi_x}. \quad (4.11)$$

These equations were used to integrate $A_{x,room,dynamic}$ and $A_{y,room,dynamic}$ twice, to determine the stride length, and the vertical displacement of the foot.

In addition, a second method of estimating velocity and displacement along the x-axis was evaluated. Because the subjects walked in (essentially) a straight line, which was (essentially) aligned with X_{room} axis, the total dynamic acceleration along X_{GS} , $A_{x,dynamic}$ was also integrated twice to estimate stride length. This corresponded to the path length traced out by the foot during walking. While this result does not correspond precisely to the room coordinates, it is a simple method of approximating these parameters of interest, and does not use the output of the y-accelerometer.

For future work with applications for the more usual case of non-straight walking (such as turning corners), the full three axes of acceleration and angular velocity will need to be analyzed and combined. In addition, it may be useful to have three two-axis accelerometer, such that each axis has two (redundant) measurements.

Integration Methods

Following determination of the dynamic components of the outputs of each of the accelerometers, a single linear integration determined¹ the velocities, and a subsequent linear integration determined the displacements, as shown in Figure 4.25 and Figure 4.27. After each integration, when the foot was flat on the floor and not accelerating, the velocity was reset to zero.

The integration bounds for both the x- and y-acceleration were determined by the characteristics of $Ax_{dynamic}$, using the z-gyroscope integration bounds as starting points. The lower integration bound was determined by starting at the first quarter-point between a pair of z-gyroscope integration bounds, and moving back in time toward the first z-gyroscope integration bound until the magnitude of $Ax_{dynamic}$ was less than 0.2 m/s^2 . If no value met this condition, the lower bound was set at the time point where the magnitude of $Ax_{dynamic}$ was a minimum, across the points between the first z-gyroscope integration bound and the first quarter-point.

The upper integration bound was determined by moving forward in time from the mid-point between two subsequent z-gyroscope integration bounds towards the second z-gyroscope integration bound until the magnitude of $Ax_{dynamic}$ was either less than 0.2 m/s^2 or the value of $Ax_{dynamic}$ was greater than 0 m/s^2 . The latter condition was for the instances where a large positive acceleration was detected. This condition, which appeared in the fourth integration bound in Figure 4.25, most likely corresponds to a strong heel-strike, so it is appropriate to cease integration, since the stride is completed upon heel strike. If neither of these conditions were met, the upper bound was set at the time point where the magnitude of $Ax_{dynamic}$ was a minimum, across the points in between the second z-gyroscope integration bound and ten points prior. Both the lower and upper integration bounds are indicated in Figure 4.25 and Figure 4.27.

1. See `accel_integrator.m` and `linear_integrator.m`, m-files available in Appendix F.2.

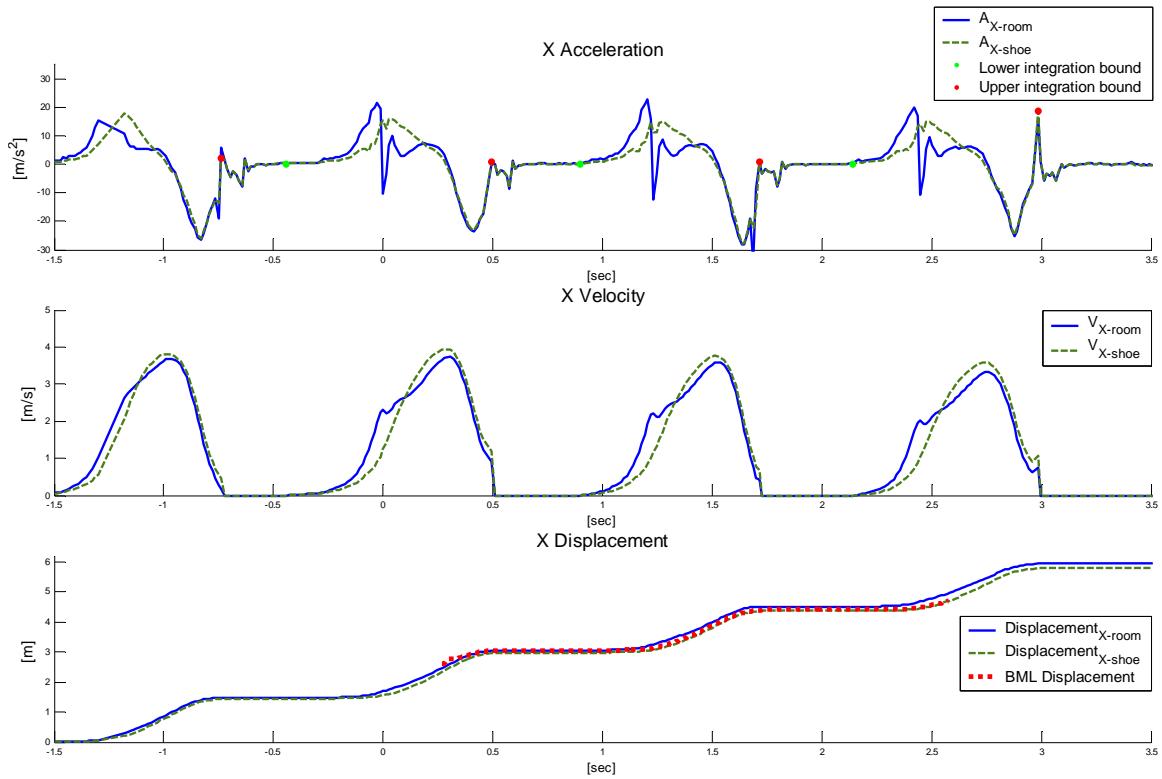


Figure 4.25 Integration of the acceleration in X_{room} and X_{shoe}

The x-displacement is shown plotted with the corresponding BML data in the lower graph in Figure 4.25. The BML data, which has a different 0 m origin than the GaitShoe, has been shifted to align with the GaitShoe results at a time point during stance (here, at approximately 0.8 sec). For evaluating the performance of the displacement calculation with the BML results (see Chapter 5), the stride lengths were compared, by calculating the difference in displacement from one stance to the next.

Figure 4.26 shows a spline fit to the linearly integrated data at timepoints corresponding to the timepoints of the BML data. In this example, the RMS error between the BML X-displacement and the GaitShoe X_{room} -displacement was 10.3 cm, while the RMS error between the BML x-displacement and the GaitShoe X_{shoe} -displacement was 14.2 cm.

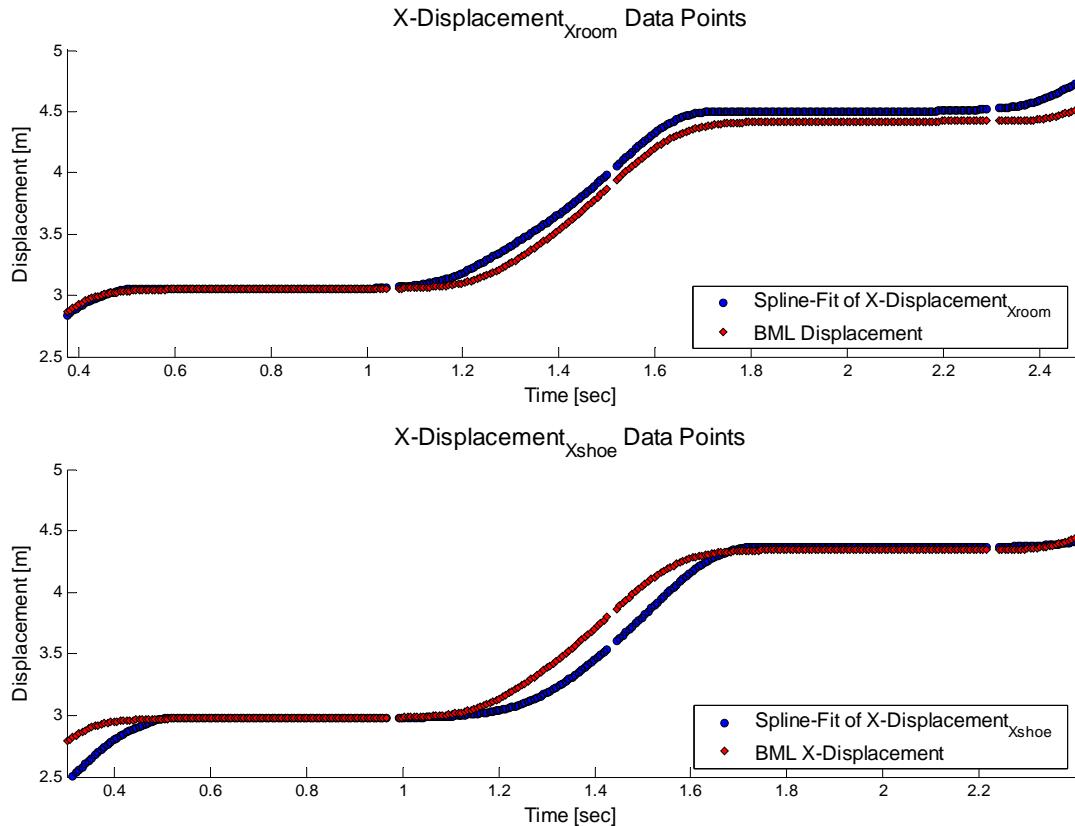


Figure 4.26 Comparison of BML X_{room} -displacement to GaitShoe X_{room} - and X_{shoe} -displacement

The y-displacement is shown plotted with the corresponding BML data in the lower graph in Figure 4.27. Since the foot LED array is mounted above the foot, during stance the value of the BML Y-displacement is on the order of 5 cm, so the BML data has been shifted to align with a displacement of 0 m a time point during stance (here, at approximately 0.8 sec). For evaluating the performance of the displacement calculation with the BML results (see Chapter 5), the peak Y-displacement and the time point at which it occurred were compared. Again, as shown in Figure 4.28, a spline was fit to the linearly integrated data at timepoints corresponding to the timepoints of the BML data. In this example, the RMS error between the BML Y-displacement and the GaitShoe Y_{room} -displacement was 3.9 cm.

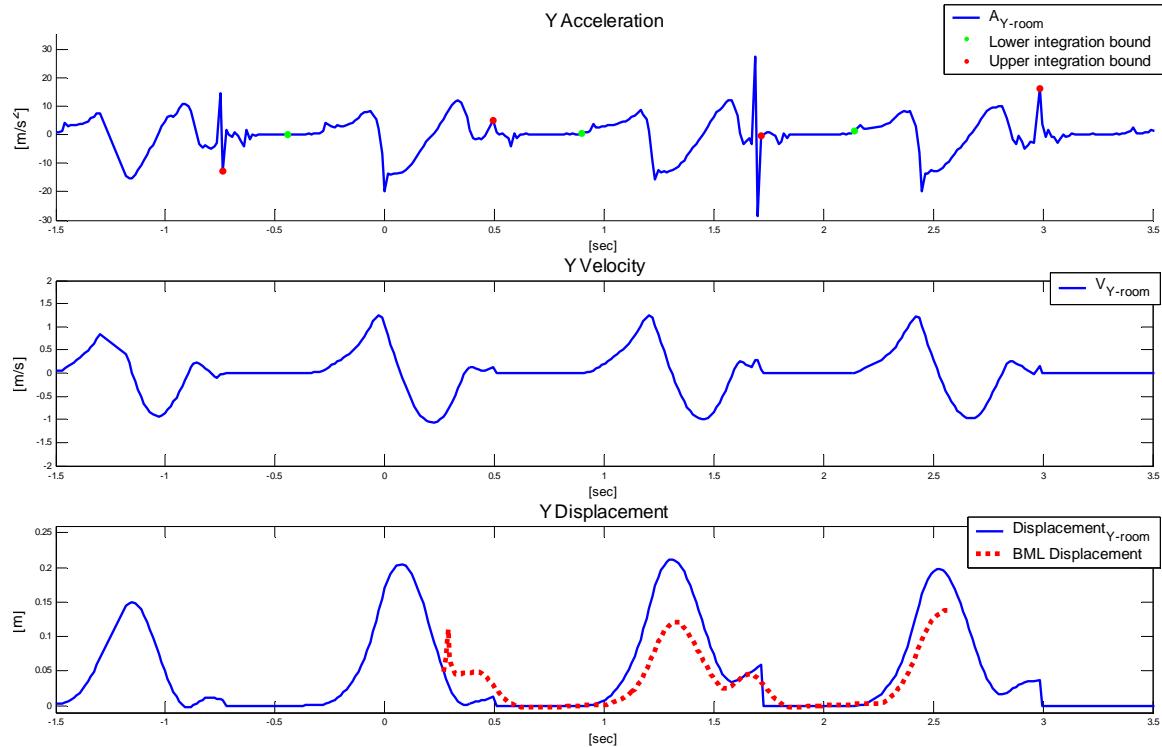


Figure 4.27 Integration of the acceleration in Y_{room}

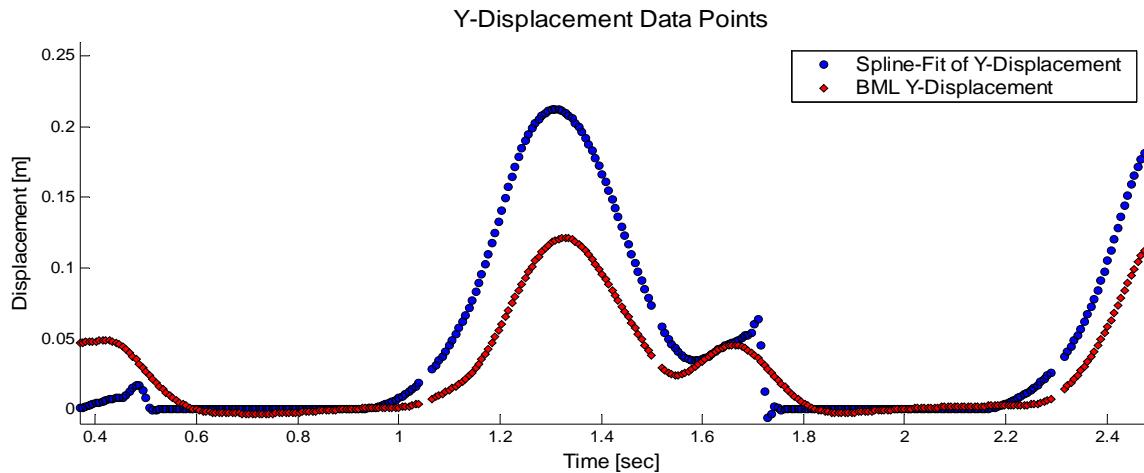


Figure 4.28 Comparison of BML Y_{room} -displacement to GaitShoe Y_{room} -displacement

The GaitShoe's X_{room} -displacement, X_{shoe} -displacement, and Y_{room} -displacement all had maximum differences compared to the corresponding BML displacements on the order of 10-15 cm. For the Y-displacement, this results in a larger percent change; the RMS error of the Y-displacement is smaller because the GaitShoe's Y_{room} -displacement is reset to

zero at each stance, which results in a large number of data points which have only a very small difference from the BML Y-displacement.

These results will likely be greatly improved by improving the GaitShoe pitch calculation, and, in addition, by incorporating all of the gyroscope and accelerometer data into the analysis.

4.5 Force Sensitive Resistors

Both types of force sensitive resistors (FSRs) were calibrated; the calibrated outputs for each shoe were added together, and the sum was used in determining heel strike and toe off timing.

4.5.1 Calibration

Use of the FSRs in the analysis required a calibration of their output to units of applied force in kg. The zero offset, representing the output of the FSR with no applied force, was not required in this analysis, because the FSRs were likely to be slightly pre-loaded during subject testing. Because the FSRs are located under the foot inside a shoe, pre-load can result when the laces are tied snugly. In addition, the equipment worn on the foot for acquisition of the MGH Biomotion Lab's data involved a large strap which wraps around the foot, which can cause further pre-loading of the FSRs. Therefore, the load on the FSRs when the foot is in the air is not expected to be 0 kg, so as long as the sensitivity is determined at small values close to 0 kg, the zero offset is not required.

Unlike the output of the accelerometers and the gyroscopes, the output of the FSRs is non-linear, due in part to the choice of conditioning electronics¹. In addition, the manufacturer's specifications report that the single part repeatability is from $\pm 2\%$ to $\pm 5\%$ of the nominal resistance. Also, over time and with use, the adhesive layer in the FSRs may break down and contribute to an increased non-linearity.

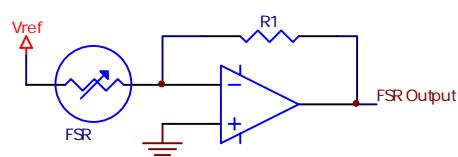
Sensitivity

The FSRs were characterized through the application of a series of forces to several FSRs. The tests were carried out using a TA-XT Texture Analyser from Stable Micro Systems, with both a 5 kg and a 30 kg load cell.

As described in Section 3.3.3, the output of the FSR can be biased if it is bent. There are two ways an FSR could be bent, either by a simple folding motion, or by being indented; the likelihood of either will depend both on the location of the FSR beneath the foot, as well as the shoe of the subject (e.g. the amount of flexibility behind the FSR). The two larger FSRs are located beneath the metatarsals, at the first and the fifth metatarsal heads. As the subject rolls off the foot preceding toe-off, the sensor portion of the FSR may experience some bending across its sensing area, particularly in flexible running shoes, which are designed to allow a great degree of flexion in the insole. Also, since the size of the metatarsal heads is on the same order as the size of the FSRs, a large amount of force applied through the metatarsal heads could result in an indentation of the FSRs into the insole, particularly if the insole is very flexible, as found in most running shoes. On the other hand, the two small FSRs are located beneath the heel pad, and are less likely to experience much bending, both because there is not typically any bending underneath the heel, and because the area of the heel pad is much larger than the sensing area of the FSRs under the heel, and so bending due to indentation is less likely.

Therefore, the output of the FSR will depend not only on the applied force, but on the method of application, which is likely to vary slightly from subject to subject, depending on the shape of their feet, and the type of shoe.

1. As described in Section 3.3.3 and Eq. 3.3, the output of a given FSR is proportional to $(R_1 + R_{FSR})/R_{FSR}$, which results in non-linear output. For future work, an implementation using the FSR at the input to a current-to-voltage converter, as shown to the right, is recommended, as the output of the FSR would be inversely proportional to R_{FSR} , with $V_{out} = - (R_1/R_{FSR}) \cdot V_{ref}$. Direct proportionality could be obtained by using R_{FSR} as the feedback resistor.



An alternate FSR implementation

To mimic these conditions during calibration, the following set-up was used: the FSRs were placed on a typical running shoe insole, which allowed some indentation, as would occur in a normal walking shoe. The applicators are shown in Figure 4.29, were wooden dowels covered with a thin layer of a compressible material, cut to the same diameter, and glued to the bottom. This was designed to mimic the metatarsal head and the calcaneous behind a layer of skin and tissue. The testing set-up is shown in Figure 4.30. The large indentor was the same diameter as the sensing area of the FSR-402. The small indentor was the same diameter as the physical area of the FSR-400; in calibrating the FSR-400, the force applied during calibration was converted to a pressure (applied across the indentor area), and converted back to the force measured across the area of the sensing area of the FSR-400.

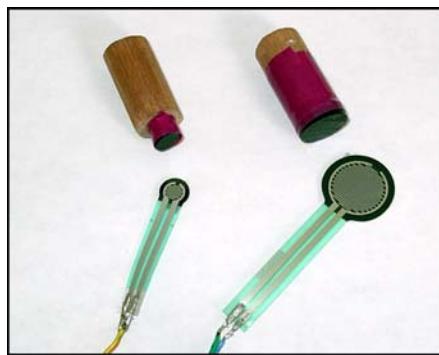


Figure 4.29 Force applicators (above) and FSRs (below)

Six of each type of FSRs (small: FSR-400, large: FSR-402, both from Interlink Electronics [77]) were tested, four which had been used in the insoles during patient testing, and two new unused sensors. The TA-XT Texture Analyser was used to apply successive forces every 0.25 kg from 0.25 kg to 10 kg, measured with a 5 kg load cell, for both the FSR-402 and the FSR-400 sensors. For the FSR-402 sensors only, the TA-XT Texture Analyser was used to apply successive forces every 1 kg from 10 kg to 30 kg, measured with a 30 kg load cell. The TA-XT Texture Analyser continually recorded the actual applied force, and was re-calibrated after each series of successive forces. Each of the successive forces applied by the TA-XT Texture Analyser, and the corresponding outputs of the FSRs were determined and correlated.

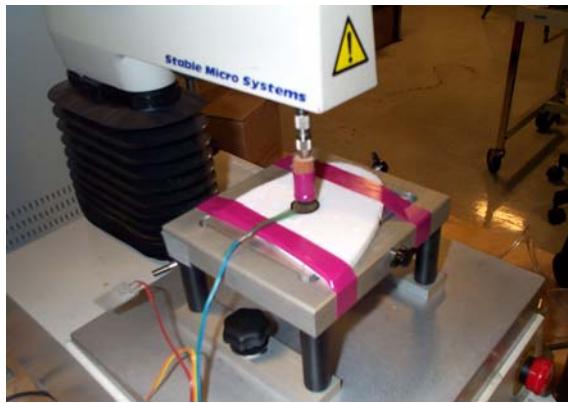


Figure 4.30 Test set-up for calibration of FSRs

Figure 4.31 shows a sample of the calibration data from a used FSR-402 sensor. The upper left graph shows the FSR-402 output during calibration, and the lower left graph shows the pressure applied by the TA-XT Texture Analyser across the indentor during calibration. The right graph shows the calibration results, with the FSR-402 output plotted vs. the applied pressure, with a second x-axis showing the equivalent applied force units.

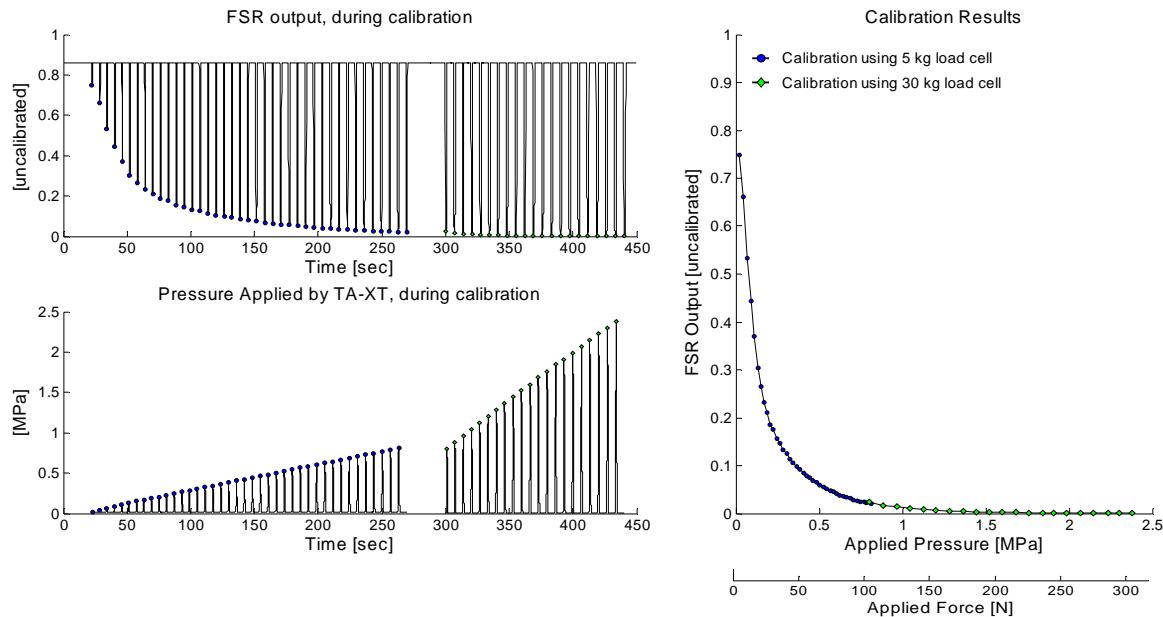


Figure 4.31 Sample FSR-402 calibration data

For each type of FSRs, a single curve was fit to the calibration data for multiple FSRs, rather than fitting individual curves for each FSR, because it was expected that the output of each FSR during calibration may not directly correspond to the output during subject

testing. As mentioned previously, FSR output may degrade over its lifetime use, and the calibration was carried out after subject testing was concluded. In addition, a few of the FSRs were replaced during subject testing, hence the replaced FSRs could not be calibrated. Though two new FSRs of each type were calibrated, the output of the new FSRs was not used for the curve fit. Since the output of all the used FSRs grouped together, despite a few FSRs being replaced during testing, it was expected that the best representation of FSR output during testing would be determined from the output of the used FSRs only. Also the FSRs are thought to have some temperature and humidity sensitivity, which could certainly affect their performance within the environment of the shoe. Finally, while the indentors were designed to mimic the conditions inside the shoe, each subject's foot is certainly unique, which may result in some minor variation in the output.

The calibration results for all six tested FSR-402s and the calibration data for the FSR-402 provided in the FSR data sheet [77], are shown in Figure 4.33. To capture the non-linear relationship between the FSR output and the applied force, first, second, and third order polynomials in exponentials were fit¹ to the calibration data from the four used FSR-402s, as shown in Figure 4.32. The third-order polynomial in an exponential was empirically found to provide a good fit to the data, and is also plotted in Figure 4.33.

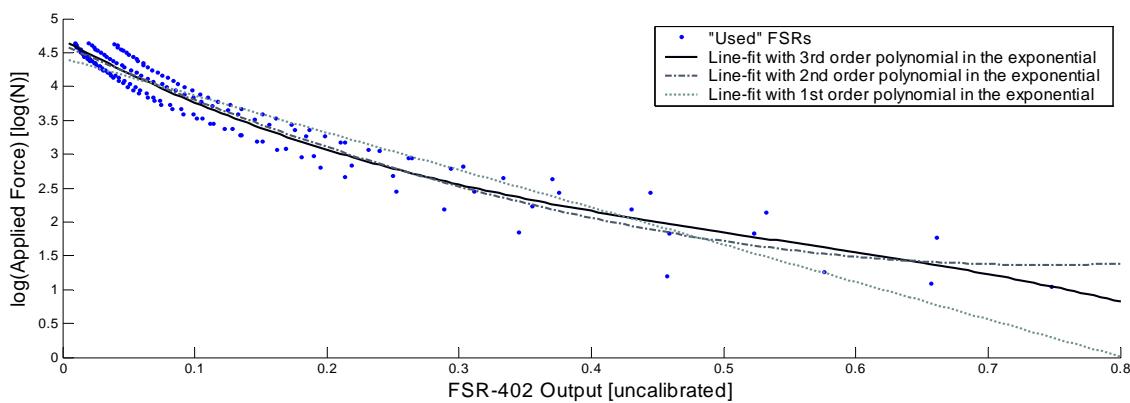


Figure 4.32 Various line-fits to the FSR-402 calibration data

- Matlab function polyfit.m was used to fit a third-order line to the applied force and the natural logarithm of the FSR output; polyfit.m returns p, polynomial coefficients, and s, for estimating the error in the line-fit.

The relationship between the FSR-402 output, V_{F402} , and the applied force in N, F , is described by Eq. 4.12.

$$F = 9.8 \cdot e^{\{-8.7 \cdot V_{F402}^3 + 14.2 \cdot V_{F402}^2 - 10.6 \cdot V_{F402} + 2.4\}} \quad (4.12)$$

Figure 4.33 also shows two additional limits: the mean value of the minimum FSR-402 output measured across all trials during subject testing (0.20), and the absolute minimum FSR-402 output over all trials (0.01). While the absolute minimum was much lower than the mean minimum output, fewer than 15% of all FSR-402 trials (145 out of 1080) had a minimum value under 0.10.

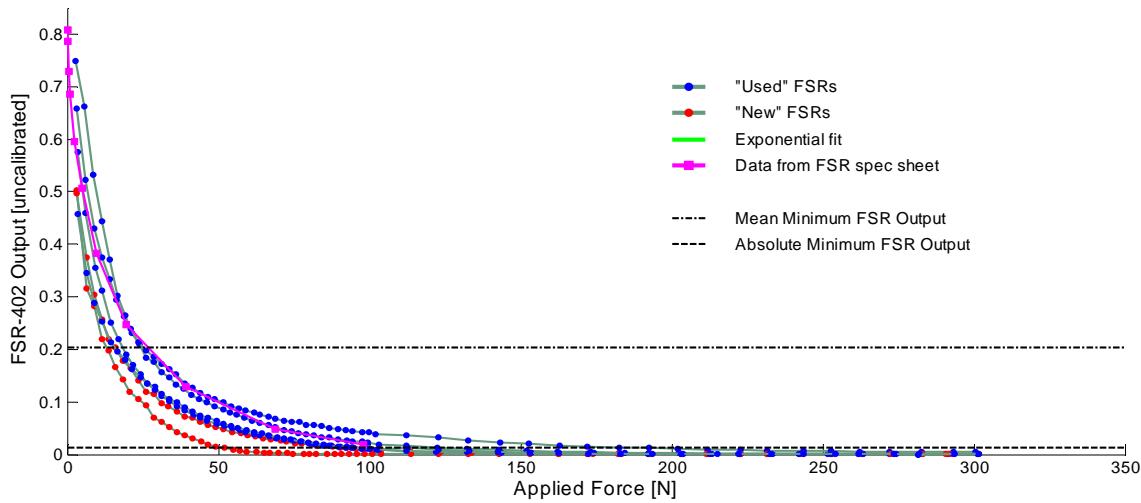


Figure 4.33 FSR-402 calibration curve

The FSR-400 calibration curves are shown in Figure 4.34. Again, four used FSRs and two new FSRs were tested (calibration data for the FSR-400 was not provided in the FSR data sheet). The compressible material on the bottom of the indentor delaminated during the testing of the last used FSRs; this FSR is shown in a different color in Figure 4.34.

As with the FSR-402s, a third order exponential was fit to the calibration data from the three FSR-400s with the indentor in the proper configuration. The resulting curve is shown in Figure 4.34, and the relationship between the FSR-400 output, V_{F400} , and the applied force in N, F , is described by Eq. 4.13.

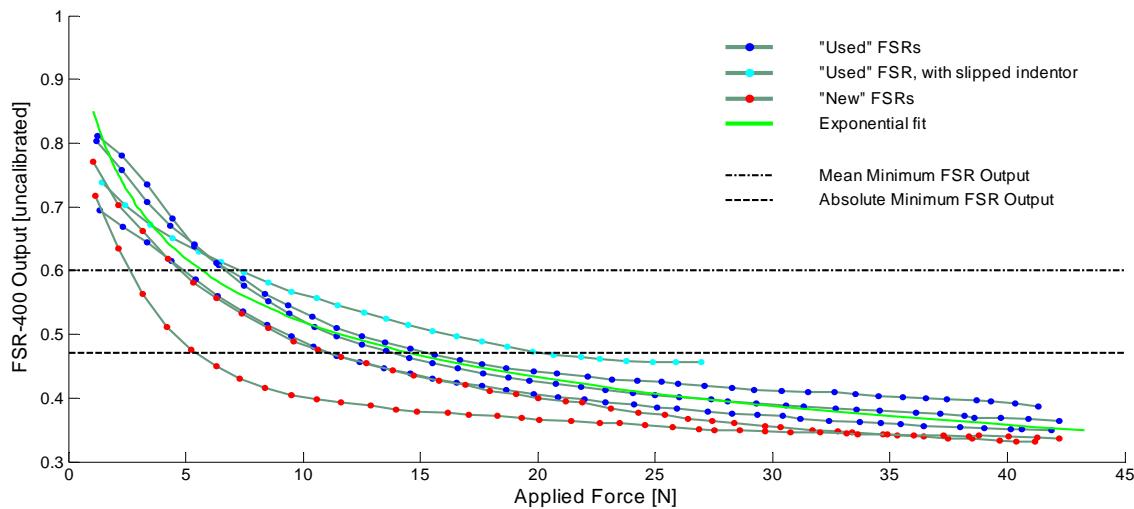


Figure 4.34 FSR-400 calibration curve

$$F = 9.8 \cdot e^{\{-10.5 \cdot V_{F400}^3 + 21.9 \cdot V_{F400}^2 - 21.6 \cdot V_{F400} + 6.8\}} \quad (4.13)$$

Again, Figure 4.34 shows two additional limits: the mean value of the minimum FSR-400 output measured across all trials during subject testing (0.60), and the absolute minimum FSR-402 output over all trials (0.47). Fewer than 2.6% of all FSR-400 trials (28 out of 1080) had a minimum value under 0.50.

Figure 4.35 shows the 95% confidence intervals, which were calculated¹ on the line-fit of the log of the applied force for the curve-fit, and converted to units of applied force in N, for both FSR types. Since the sensitivity curve is an exponential, it only approaches zero, but as discussed above, the FSRs are likely to have a small amount of pre-load. The calibration started at 0.25 kg, to define the output when a small force was applied.

The 95% confidence intervals were calculated for each FSR from the absolute minimum value seen during testing (0.47 for the FSR-400, 0.01 for the FSR-402) to 0.8, with an increment of 0.005; the mean across these values was 14.64 N for the FSR-400, and 14.95 kg for the FSR-402.

1. Matlab function polyval.m was used to obtain error estimates, using "s", a structure returned from polyfit.m for estimating the error in the line-fit.

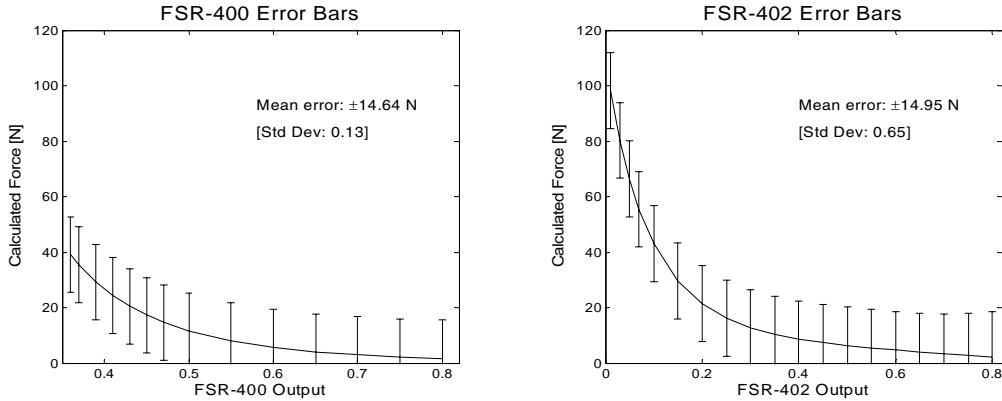


Figure 4.35 95% confidence intervals for FSR sensitivity curves

This error is rather large, and is likely due at least in part to the non-linearity of the FSR response. Different conditioning electronics, such as those discussed at the beginning of this section, that would result in a more linear FSR response should be implemented before using the FSRs for numerical analysis of the force distribution. In the analysis for this work, the FSRs were calibrated and summed together for use in determining heel strike and toe off timing.

4.6 Bend Sensors

4.6.1 Calibration

For the bend sensors, two types of information were required: the zero offset, to center the output around zero; and, the sensitivity, to convert the output to units of degrees ($^{\circ}$).

Zero Offset

The zero offset of the bend sensors was simply the output of each bi-directional bend sensor while lying flat; the zero offsets are summarized in Table 4.5.

Sensitivity

Each bend sensor was characterized individually; the native resistance of the individual bend sensors varies widely, making the output of each pair of bend sensors unique. The

test setup is shown in Figure 4.36. Pairs of 9 V batteries were taped together and used as a weighted clamp to position the bend sensors. A printed graph with lines radiating to mark angles from -50° to $+50^\circ$ was taped to the table to guide the calibration process. One pair of batteries was taped to the table such that they created a gap parallel to the 0° line, with the right edge aligned with the $(0,0)$ mark on the graph. The bend sensor was threaded through this pair of batteries, and positioned such that the midpoint of the bend sensor was aligned with the $(0,0)$ mark on the graph. A second pair of batteries was placed over the bend sensor, about an inch away from the pivot point, and was used for positioning the bend sensor.

For each of the four bend sensors, a static calibration test was carried out, with data collected at 10° increments from 0 to $+50^\circ$, and from 0 to -50° . First, the bend sensor was aligned at the increment, and a small weighted box was place on each side of the positioning pair of batteries, to hold the bend sensor securely in place. One of the FSRs was pressed to indicate the start of data collection, and after 20-30 seconds, a second FSR was pressed to indicate the end of data collection. Then, the bend sensor was moved to the next increment, and the process was repeated.

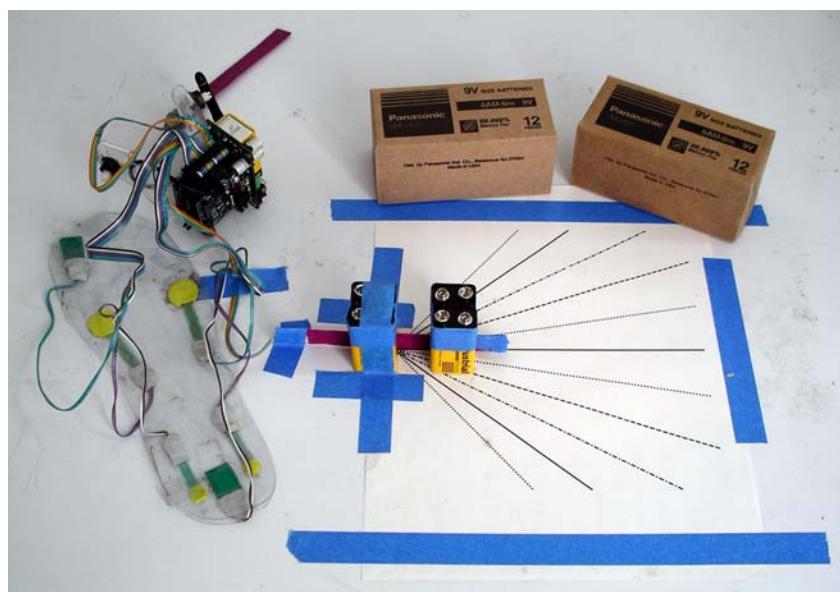


Figure 4.36 Method of calibration of bend sensors

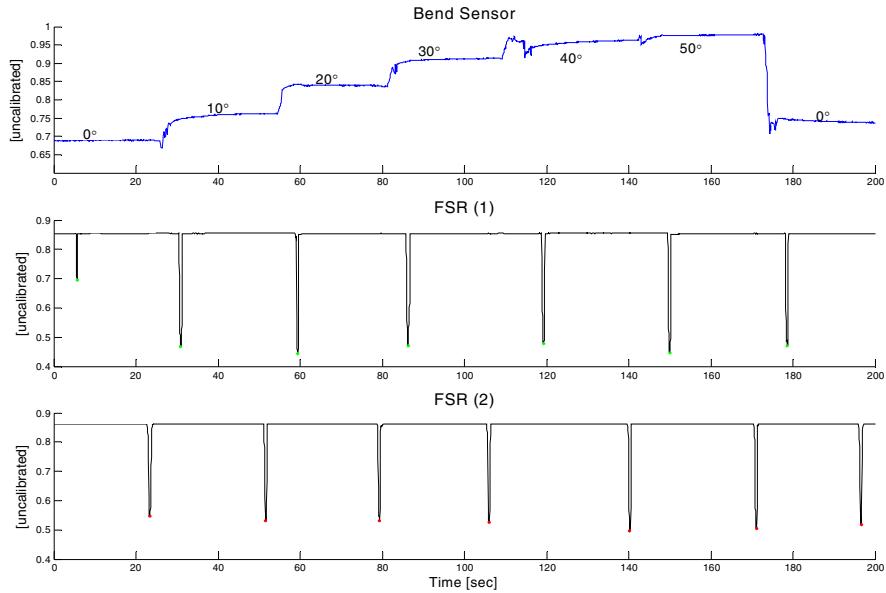


Figure 4.37 Data from calibration of sensitivity of the bend sensor

A sample of the data from the calibration of one of the bend sensors is shown in Figure 4.37 (the FSR plots are included to indicate the times of data collection; rapid changes in the bend sensor output, such as seen around 115 sec corresponds repositioning of the bend sensor in between the data collection). The output appears to change linearly from 0° through +30°, but at angles higher than 30°, the output nears the maximum output of 1, and starts to saturate. In addition, there is some drift present: the output at 0° at the end of the calibration trial is different than at the start (a slight change in positioning could contribute to less than 1° of this change, but the full change is closer to approximately 5°, suggesting another process is contributing). In particular, it is likely that the electrical tape used to hold the two uni-directional bend sensors is contributing to the drift as it stretches and relaxes, and in addition, the bend sensor output is likely to be affected by the radius of curvature at the bending point.

For each bend sensor, the clipped means¹ were calculated on the output at each of the measured different angles, and on the data from the 0° position at the start of each trial (the data from the 0° position at the end of each trial was not used). A straight line was fit² to

1. Again, as discussed in Section 4.3.1, spurious points were eliminated by using the clipped mean.

2. Matlab function polyfit.m was used.

the clipped mean data points; for data which saturated at the upper limit, only the first clipped mean value greater than 0.95 was used, so as not to skew the line-fit away from the non-saturated data (the calibration was carried out only once¹). The clipped mean points, and the resulting line-fit are both shown in Figure 4.38 for all four sensors. The sensitivity (slope of the line-fit) and the zero offset are shown in Table 4.5.

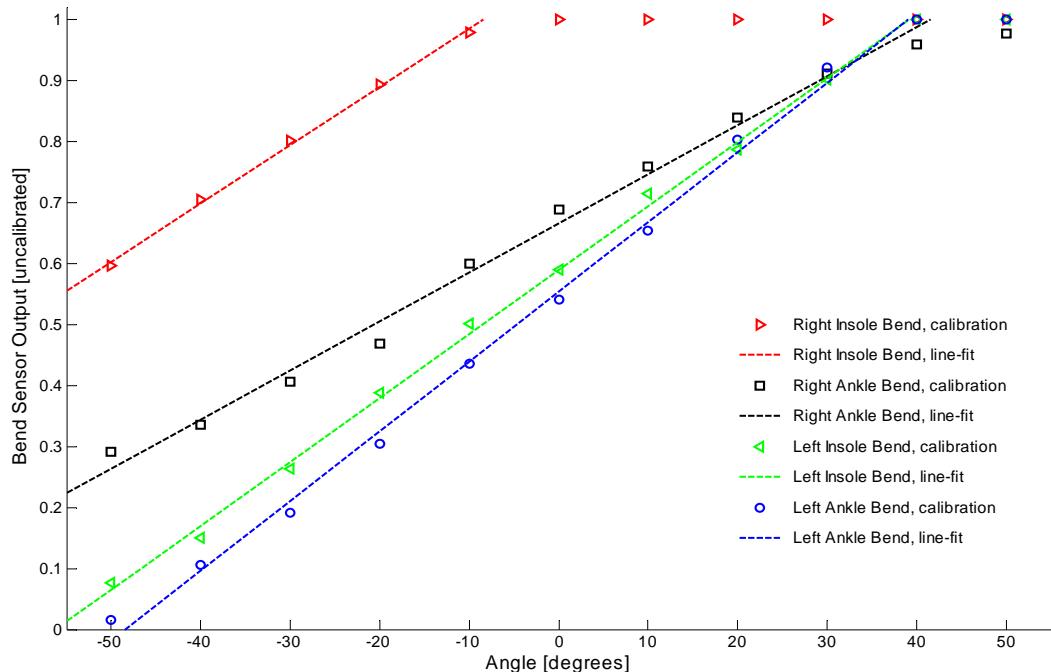


Figure 4.38 Line fit for determining the sensitivity of bend sensors

One sensor (shown in red in Figure 4.38) actually had a uni-directional output, due to a deficient individual bend sensor, and saturated at the upper bound. This sensor was located in the right insole, and the saturation of this sensor was not an issue during subject testing of normal gait, as flexion of the insole is only possible in one direction; negative bend corresponded to insole flexion. The left insole sensor also had some saturation, but only at the higher range of positive bend, so it was not an issue during subject testing either.

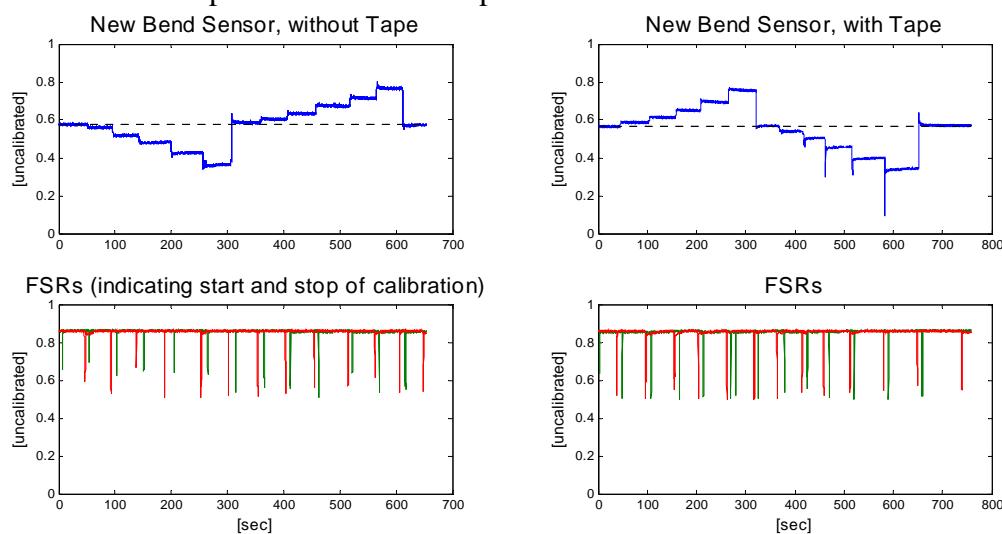
1. As the rest of this section discusses, the bend sensor did not perform well; though the calibration would be more accurate if the bending routine was repeated multiple times, the inaccuracies in the bend sensor discussed below in Section 4.6.2 and Section 4.6.3 were likely more influenced by the placement of the bend sensor and the drift issue discussed above. Future work with the GaitShoe should briefly investigate whether the influence of the placement and the drift can be alleviated, and if so, a better calibration routine should be devised at that time. But it is likely that far better results would be achieved by switching to a different type of sensor, or by placing a second IMU on the shin.

TABLE 4.5 Bend sensor sensitivities and zero offsets

Sensor	Sensitivity [degrees/output]	Zero Offset [output]
Left, Ankle	87.7	0.54
Left, Insole	95.4	0.59
Right, Ankle	124.5	0.39
Right, Insole	104.8	1.00

The bend of the ankle bend sensors corresponds to plantar flexion and dorsiflexion; positive bend corresponded to dorsiflexion and negative bend corresponded to plantar flexion. The two sensors located at the left and right ankles also had some saturation, however, the outputs of both sensors are linear across the range of typical angles encountered. A healthy gait cycle typically has a maximum dorsiflexion close to 10° and a maximum plantar flexion of 10-20° [105].

To investigate the issue of drift further, the calibration was run on a bend sensor with two brand new uni-directional bend sensors, with and without tape covering. The results are shown in Figure 4.40; a black dashed line has been plotted over the bend sensors, corresponding to the value of the bend sensor at time zero. Again, the FSRs were used to indicate the start and stop of each calibration point.

**Figure 4.39** Bend calibration routine, on a new sensor with and without tape

These results do not show the drift present in the earlier calibration plots, indicating that the drift is likely to be due to aging of either or both the bend sensor and the electrical tape used to protect the bend sensors and to force the two uni-directional bend sensors to follow the same motion (the bend sensors used in the subject testing were assembled with the tape in January 2003 at the latest). Further work is necessary to determine which aging process is the major contributor to the drift. If the electric tape, other methods of protecting the bend sensors (and forcing the same motion) could easily be found, such as slipping a plastic sleeve over the sensors. However, if the bend sensor itself is degrading as it ages, the use of this part may need to be rethought as it would be inconvenient and expensive (each individual bend sensor is \$10, for a total cost of \$80 across both feet) to replace.

4.6.2 Placement of the Bend Sensors

One of the bend sensors was used to check the variation in the output due to a change in the location of the pivot point, as shown in Figure 4.40. The insole bend sensors were not repositioned before each subject, but were located such that their midpoints were roughly aligned with the metatarsals, since flexion in the insole occurs about the metatarsals. The ankle bend sensors were positioned between the heel of the foot and the back of the shoe such that the midpoint was roughly aligned with the back of the shoe. In normal use, the ankle attachment (shown in Figure 3.9) would be mounted on the subject's ankle to hold the ankle bend sensor in place. However, testing was carried out with the subject simultaneously wearing the MGH Biomotion Lab equipment, and so as not to unduly interfere with the gait of the subject, the ankle attachment was mounted on the MGH Biomotion Lab shin attachment rather than directly against the ankle. This resulted in the ankle attachment being located further away from the edge of the shoe, particularly on taller subjects, so the bend sensor had to be pulled out of the shoe in order to be enclosed completely by the ankle attachment. This resulted in the bend sensor pivoting about a point between the end of the sensor and the midpoint.

Figure 4.40 demonstrates that the sensitivity of the output remains similar over the three different pivot points: the midpoint, and the 1/4 and 3/4 points along the sensor's length. However, there is some variation in the zero offset value, which is likely due to the drift seen in Figure 4.37, as these tests were performed in succession.

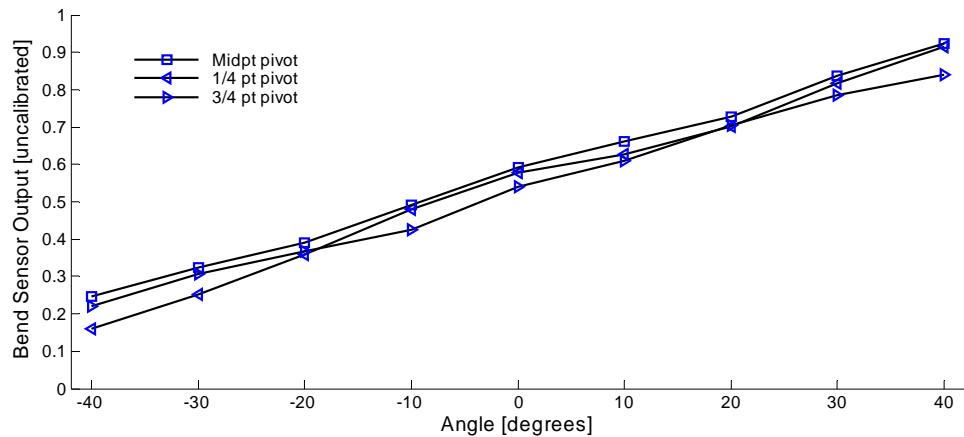


Figure 4.40 Bend sensor, calibrated with different pivot points

4.6.3 Plantar Flexion and Dorsiflexion

The bend sensor at the ankle was used to measure the angle of plantar flexion and dorsiflexion. However, as discussed above, the bend sensors have a baseline drift in the output. In addition, the ankle attachment, which was designed to hold the bend sensor next to the ankle to prevent buckling, was placed higher relative to the back of the shoe, during subject testing. This resulted in the ankle bend sensor being more likely to buckle during data collection. Also, plantar flexion and dorsiflexion were determined from the BML data from the pitch of the foot array and the pitch of the ankle array, resulting in the angle between the shank and the foot. There are two difficulties in using the BML data to validate the calibrated bend sensor data. First, the shank and foot arrays are visible to the camera detection system at different (but overlapping) times, which results in a shortened amount of data available as only the overlapping data can be used to determine the pitch. The second difficulty is that the BML data occasionally contains outlying data points; because two different data vectors are combined, there are two sources of error: these must be removed, and thus further shorten the data available for comparison.

The bend sensor output for an ankle bend sensor, and the combined output of the BML foot pitch and shank pitch, resulting in plantar flexion / dorsiflexion curve measurement for the corresponding foot are shown in Figure 4.41 (the bend sensor data shown is the result of a spline fit to the bend sensor calibrated output, using the time points corresponding to the BML data).

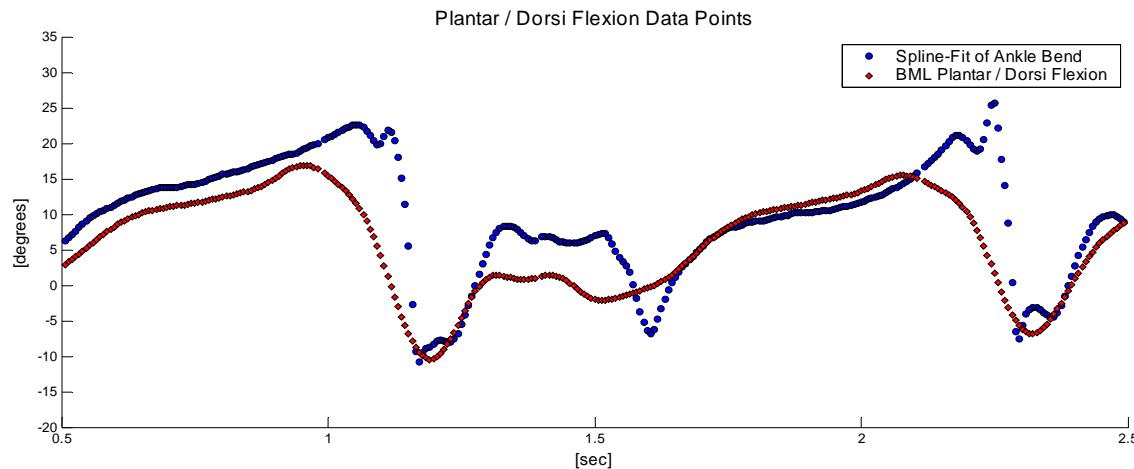


Figure 4.41 Plantar Flexion and Dorsiflexion from the GaitShoe (left) and the BML (right)

Although the shape of the calibrated bend sensor output generally follows the shape of the BML plantar flexion / dorsiflexion curve, the RMS error between the two curves is 6.3° . With a full range across the BML plantar flexion / dorsiflexion of less than 30° , this represents an error of more than 20%. The sharp changes in the GaitShoe ankle bend output (e.g. shortly after 1 sec and around 2.25 sec) are most likely due to the sensor buckling as a result of the poor location of the shoe attachment, as the direction of flexion reverses.

A sample of the calibrated insole bend data is shown in Figure 4.42, for the same gait trial as the calibrated ankle bend data shown in Figure 4.41. The minima correspond to the foot motion preceding toe off. This is a parameter that the BML does not measure: the BML foot array is mounted in the center of the foot and cannot distinguish between pure rotation of the foot, and flexion of the foot with the forefoot flat on the ground.

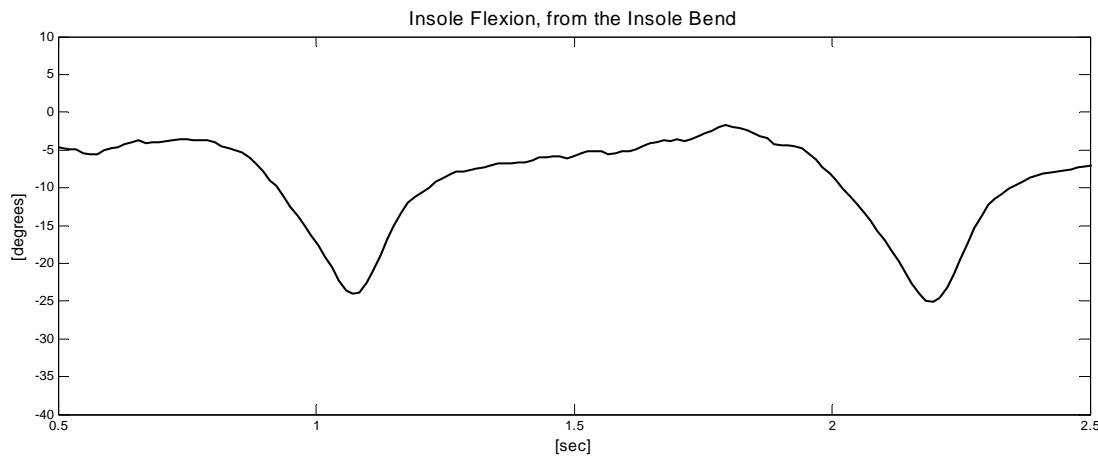


Figure 4.42 Calibrated insole bend sensor output

Because of these issues, the bend sensor signals were not further validated with the BML data. Future work with the GaitShoe using this bend sensor should involve a different design of the ankle attachment, or evaluation with a system that allows the bend sensor to be held next to the ankle as it would under conditions where only the GaitShoe is collecting data. Alternatively, other methods of determining plantar flexion / dorsiflexion should be considered, such as the use of fiber optic sensors, or placement of a second IMU on the ankle, which would provide the full gyroscope and accelerometer information, and would allow the position of the shin and foot to each be determined independently, and then combined to find relative positions, such as plantar flexion / dorsiflexion.

4.7 PVDF Strips

The polyvinylidene fluoride (PVDF) strips were included for their rapid dynamic response, and were expected to be useful in determining heel strike and toe off timing. Unfortunately, the output varied greatly, not only from subject to subject, but between the left and right feet of a given subject, and even from step to step within a given subject. The PVDF outputs from five females with healthy gait are shown in Figure 4.43; data from the left shoe are on the left, and data from the right shoe are on the right, and the outputs from the PVDF sensors at both the heel and the great toe are included. The raw PVDF output

was scaled from the twelve bit output such that the output ranged from 0 to 1, and was centered around zero.

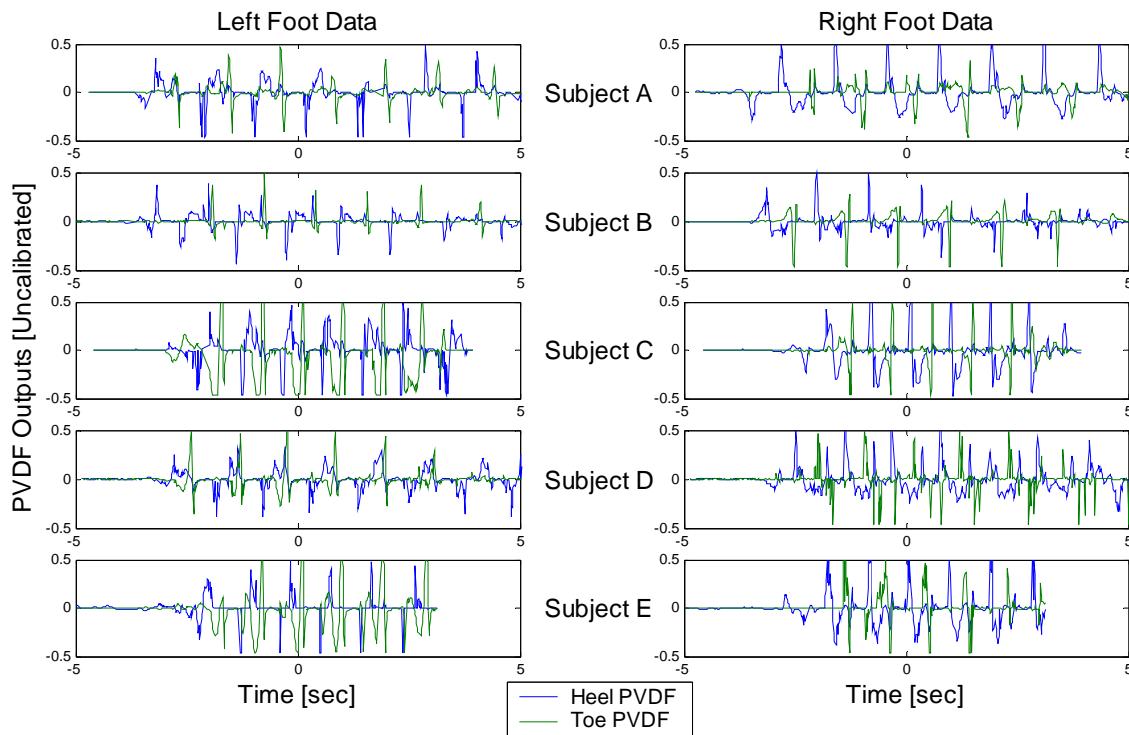


Figure 4.43 Various PVDF output plots

Some of the variation seen across Figure 4.43 may be caused by the physical design of the shoe worn by the subject: the PVDF strip is traditionally used to measure vibration, and the subjects were asked to wear running shoes, which are designed to minimize impact on the foot via compressible components; in addition, the PVDF strip output is likely sensitive to motion of the foot within the shoe. Thus, some variation may be due to the PVDF strip picking up the dynamics of shoe as well as the dynamics of the foot moving within the shoe, rather than the dynamics of gait. To investigate the effect of the shoe type, data were collected on the same subject wearing three different types of shoes: a running shoe (manufactured by Reebok), a walking shoe (manufactured by Ecco), and a stiff clog (manufactured by Dansko); the shoes are shown in Figure 4.44.



Figure 4.44 Reebok running shoe, Ecco walking shoe, and Dansko clog.

Figure 4.45 shows the uncalibrated¹ data for each of three shoe types. The data are all from the right foot, and the same insole was used in all three trials. The back edge of the insole was aligned with the back of each shoe and taped into place, to position the PVDF strips in the same location relative to the subject's foot. Visually, the PVDF strips have a stronger similarity within a shoe-type than between shoe-types, particularly the toe PVDF strips, suggesting that the type of shoe worn during testing may indeed contribute to changes in the PVDF output.



Figure 4.45 Comparison of PVDF output in three different types of shoes

1. To display both PVDF output clearly on one graph, the PVDF-heel output has been shifted by a value of one.

The PVDF strips were selected to be placed at the toe and the heel were because they were expected to have a faster response than the FSRs. Figure 4.46 shows the uncalibrated (centered around zero) PVDF outputs and the sum of the calibrated FSR outputs for comparison, as well as the heel strike time and toe off time determined by the BML. The heel PVDF output does appear to rise slightly in value one time step before the FSRsum output rises (and before the BML heel strike time); however, the PVDF sensor is located centrally beneath the heel, and closer to the rear edge of the heel, while the FSRs at the heel are located medially and laterally, so this very slight anticipation by the PVDF sensor may be due only to a better location within the insole. In addition, as seen in Figure 4.43 and Figure 4.45, the heel PVDF output tends to have multiple peaks, due both the dynamic nature of the PVDF output, as well as the likely response to movement of the foot within the shoe, whereas the FSRsum has a direct correspondence to the actual force distribution beneath the foot. The toe PVDF output has already returned to the baseline at the BML toe-off time, making it no more useful than the FSRsum.

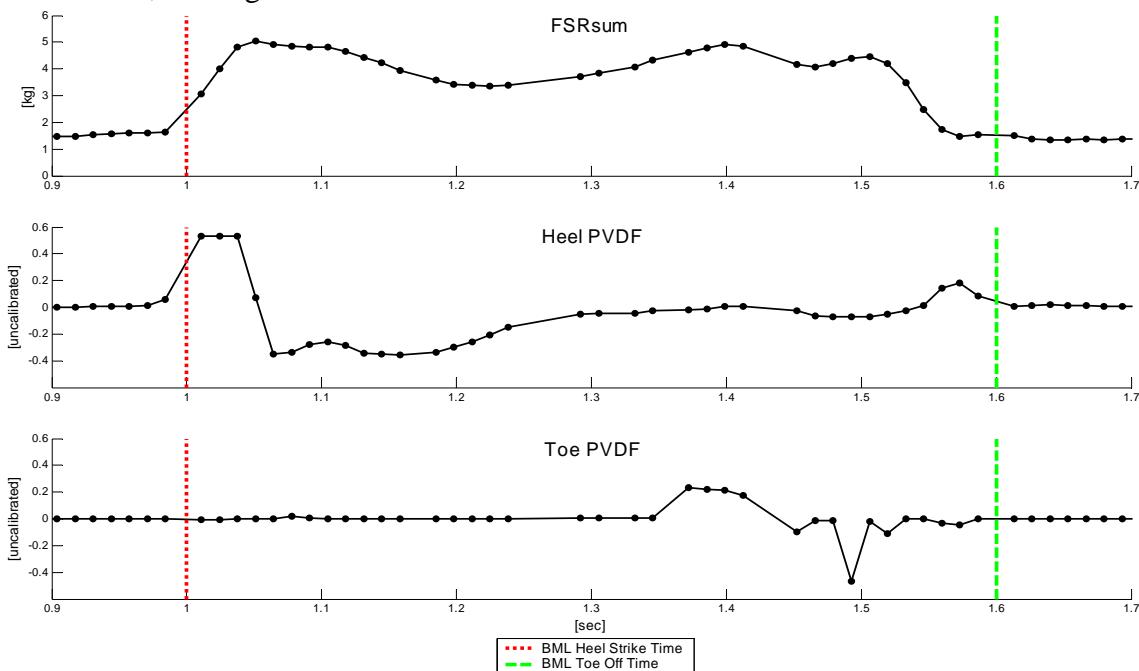


Figure 4.46 Comparison of PVDF response to FSRsum response

This PVDF output certainly contains interesting information, and, the potential change in response in different types of shoes may be useful for future work in evaluating orthotics

or different types of shoes for a given patient. However, it was selected for this work for use in determining heel strike and toe off; the FSRs appear to quite useful, not only in determining heel strike and toe off, but in providing information about the force distribution underneath the foot. The analysis of heel strike and toe off time, as described, did not employ the use of the PVDF output, but rather relied on the FSRsum, and information from the accelerometers and gyroscopes.

For future work with the GaitShoe, the PVDF strips should be replaced with FSRs, which will allow the FSRs to be placed directly under the heel pad and under the great toe, resulting in a more complete force distribution profile, and possibly improving the determination of heel strike and toe off times.

4.8 Electric Field Sensor

The electric field sensor was developed after subject testing had started, so it was included during the testing of the last five subjects to investigate its utility for measuring the height of the foot above the floor. The purpose was both to evaluate the implementation, as well as to investigate the signal output. Electrodes for the electric field sensor were created by placing copper tape on the bottom of the shoe. Figure 4.47 shows the uncalibrated output of the electric field sensor on five different subjects¹. The output level which corresponds to the zero height at the initiation of gait (e.g. the output while the subject stands with both feet flat on the floor, before starting the gait trial, around time -6 sec in the graphs in Figure 4.47) is indicated.

The physical implementation required placing copper tape on the bottom of the shoe to construct the sensor, and co-axial cable to connect the sensor to the circuit board (the driven shield signal was connected to the outer layer of the co-axial cable). The copper tape on the bottom of the shoe lasted throughout the each subject's testing. The co-axial

1. These subjects were the five subjects on which the electric field sensor was used; data were not collected from the right foot sensor on subjects A and B; these subjects are not the same as those in Figure 4.43, above

cable has a diameter of 3 mm; for most athletic shoes, the co-axial cable can be placed in a tread groove to avoid adding this small thickness under one side of the shoe, however, this connection may need to be redesigned for use with other types of footwear.

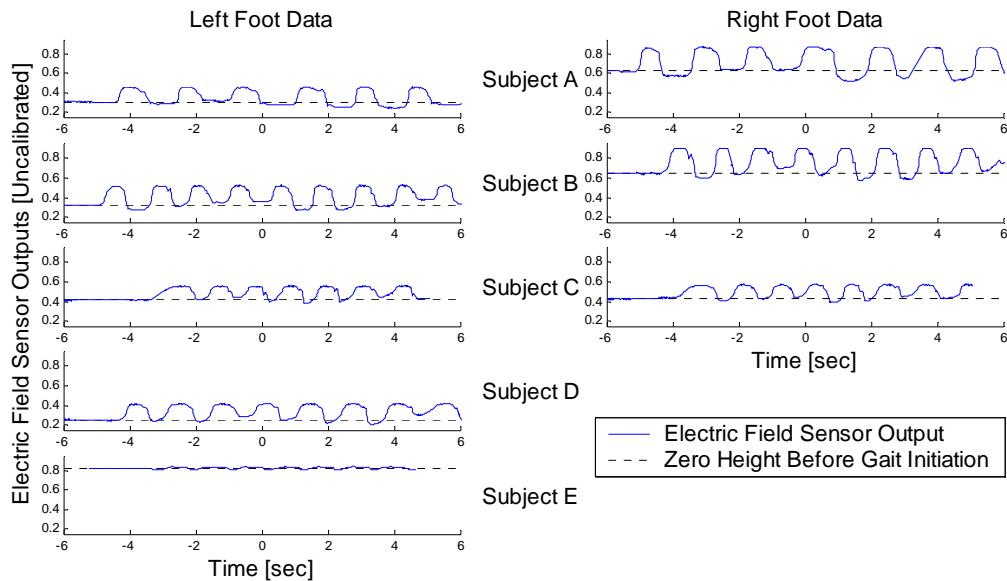


Figure 4.47 Various electric field sensor output plots

In addition, as seen in Figure 4.47, the output of the electric field sensor was greatly dependent on the implementation of the electrodes, which varied depending upon the size of the subject's shoe. the range of the output varies greatly over all eight implementations. Also, the actual output of the electric field sensor is likely to vary depending on the construction of the floor. Interestingly, most subjects generally pass the force plate between +1 sec and +4 sec; in the graphs shown in Figure 4.47, the electric field sensor output generally dips below its initial zero height during this time, as the metal force plate provides increased capacitive loading.

Figure 4.48 presents data collected during one of the gait trials from Subject A. The top graphs show the height of each foot, as measured by the MGH Biomotion Lab (BML). The height is measured by the foot array, which is located above the foot; thus, the height is never 0 cm, because the lowest height seen during stance (approximately 10 cm on the left foot, and approximately 5 cm on the right foot) corresponds to the actual height of the

array above the floor. The bottom graphs show the output of the electric field sensors on the left and right feet. The electric field output reaches its maximum value once the foot is lifted above the floor to a distance where the sensor can no longer couple into the floor. By visual comparison with the BML height graphs, both electric field sensors appear to have a range up to approximately 3-5 cm, over a third of the output range. The electric field sensor was located on the heel of the shoe, while the foot arrays were located mid-foot; this results in the electric field sensors anticipating the changes seen by the foot arrays.

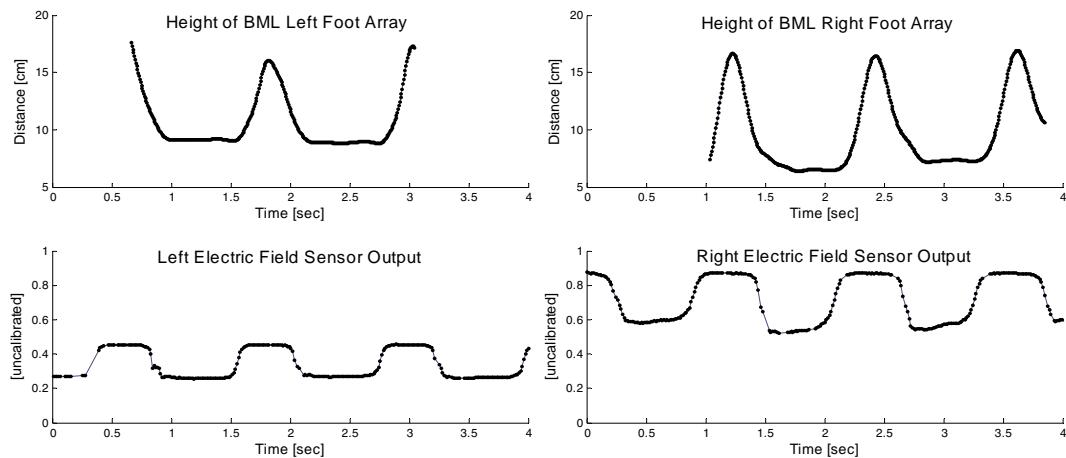


Figure 4.48 Comparison of the electric field sensor output to foot height.

Before using the electric field sensor as a method of determining the height of the foot above the floor, further investigation into the design of the electrode is needed. For instance, as discussed in Section 3.3.7, after subject testing was completed, a second design was developed, using a round electrode made out of copper tape, with the driven shield inside the shoe. This sensor was not tested at the Biomotion Lab, but it does appear to have a range greater than 5 cm. In addition, two electrodes can be used underneath the foot, to provide measurements of height at two different locations. The output of the two electrodes on the left and right feet are shown in Figure 4.49. Again, the value of the output depends on the electrode configuration. The sensors located at the heels both clearly measure a decrease in distance above the ground before the sensors located at the toes (e.g. at 1.75 sec in the left foot heel sensor, and at 2.4 sec in the right foot heel sensor), while the sensors located at the toes are clearly measuring an increase in distance above

the ground after the sensors located at the toes (e.g. at 1 sec in the left foot toe sensor, and at 1.75 sec in the right foot toe sensor).

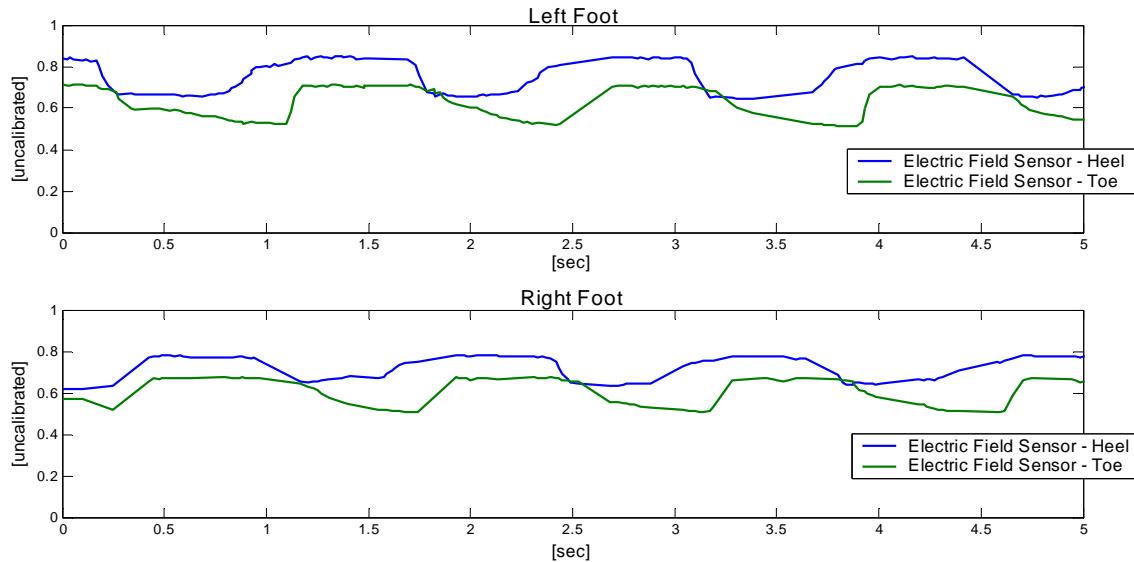


Figure 4.49 Sample output from the second electrode design

Once the electrode design has been finalized, methods of preparing the electrode such that it is consistent from subject to subject should be investigated. For instance, rather than affixing copper tape, electrodes could be pre-cut from a thin metal foil, which would be taped to the bottom of the shoe. Finally, data should be collected on different floor types, from wood floors and sidewalks to concrete floors with and without rebar, to generate a family of calibration curves and allow calibration of the electric field sensor signal.

4.9 Heel Strike and Toe Off Timing

Timing of heel strike and toe off were determined using several of the GaitShoe sensors. Initial values for each were determined using the sum of the four FSRs, "FSRsum", which was calculated when the FSRs were calibrated, as described in Section 4.1.4. These initial values were used to determine the integration bounds for the z-gyroscope. The final values of the toe off timing were determined from the pitch, and alternative values for the heel strike timing were determined from the x-accelerometer integration bounds.

FSRsum Determination of Heel Strike and Toe Off Timing

The four FSRs were coarsely distributed underneath the foot, with two underneath the heel, medially and laterally, one underneath the first metatarsal head, and one underneath the fifth metatarsal head. Though this cannot provide a complete picture of the force distribution underneath the foot, the information can still be of use.

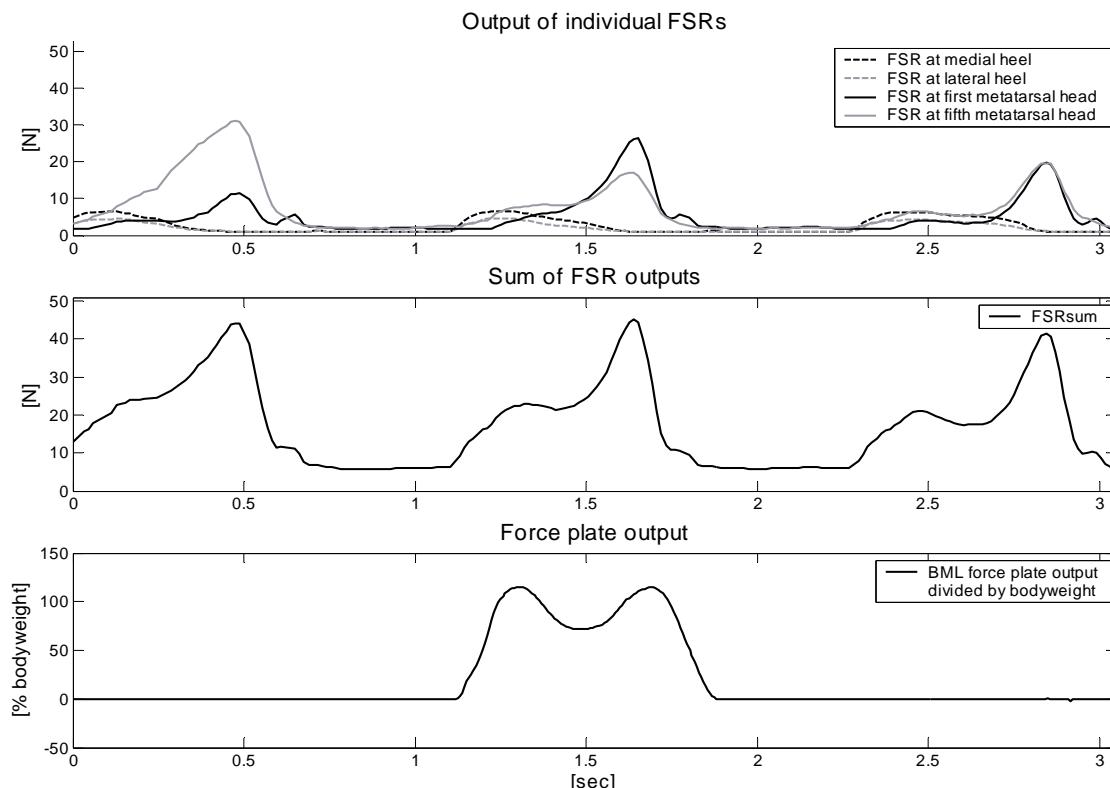


Figure 4.50 Comparison of the sum of the four calibrated [N] FSR outputs to the force plate output

The calibrated FSR output with units of force [N] (e.g. the total force applied across the FSR), the sum of the four calibrated FSR outputs, and the BML force plate output are shown in Figure 4.50. The FSRs only cover a small percentage of the total weight-bearing area underneath the foot; as such, they only measure a portion of the total force. Thus the shape of the summed FSR output in the middle graph is different than the bottom graph of the force plate measurement, because the summed FSR output is subsumed by the total force output.

For further comparison, data collected using the Tekscan F-Scan system [32] on two different subjects was examined. One subject had a peak pressure under the heel area of 32 psi, a peak pressure under the area of the first metatarsal head of 52 psi, and a body weight of 144.8 pounds, while the other subject had a peak pressure under the heel area of 43 psi, a peak pressure under the area of the first metatarsal head of 33 psi, and a body weight of 162.6 pounds [92]. These are similar ranges as the calibrated FSR data, shown in Figure 4.51 with units of pressure [psi] (e.g. the pressure applied across the FSR sensing area), though Figure 4.51 presents data from a subject with a body weight of 253 pounds.

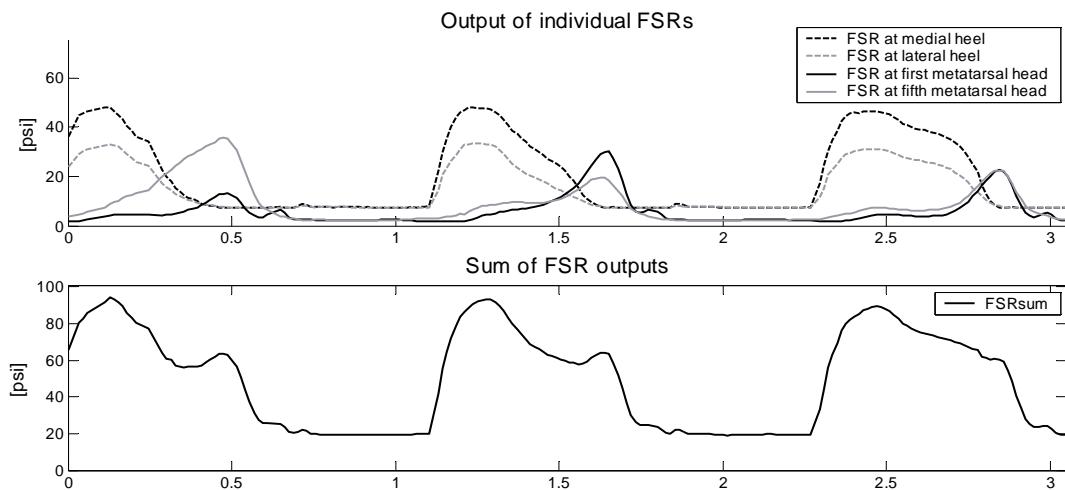


Figure 4.51 Comparison of the sum of the four calibrated FSR outputs in pressure units of psi.

Figure 4.50 demonstrates that the larger FSR-402s (located under the metatarsals) measure more force, while Figure 4.51 shows that even though the smaller FSR-400s (located under the heels) measure less force, the pressure under the heel, for this subject, is higher than underneath the metatarsals.

The change in the shape of the FSRsum, with either force or pressure units, as compared to the typical "m"-shape measured by force plate is a result of the very discrete measurements taken by the four FSRs. However, this does not render the FSR measurements useless: in both Figure 4.50 and Figure 4.51, the shape of the FSRsum is similar across the three steps shown, however, there were substantial differences in the weight distribution between the first and fifth metatarsal heads. In the first step shown, more weight is on the

fifth metatarsal, in the second step, more weight is on the first metatarsal, and in the third step, the weight is distributed fairly evenly across the first and fifth metatarsal heads. This is information which cannot be obtained from a standard force plate.

In the BML, timing of heel strike and toe off are determined by examining the force plate output; when it first crosses a certain threshold, that time point is identified as heel strike, and when it re-crosses that threshold, that time point is identified as toe off.

This general approach¹ was adopted for deriving the heel strike and toe off times from the FSRsum, as shown in Figure 4.52. In order to improve the resolution, a spline was fit to the FSRsum with time points every 1 msec; the ideal data rate of the GaitShoe was 75 Hz, which corresponds to a complete data packet every 13.4 msec. The "no-load cutoff" was calculated from the mean of the minimum value of the FSRsum and the clipped mean of the FSRsum. The no-load cutoff was used as the limit below which the FSRsum value could correspond to a no load situation, such as during swing. The times of the FSRsum points (original data, not the spline-fit) that occurred just before and just after ($\text{FSRsum}_{\text{no-load-}}$ and $\text{FSRsum}_{\text{no-load+}}$, respectively) the FSRsum crossed the no-load limit were identified; these are marked in Figure 4.52.

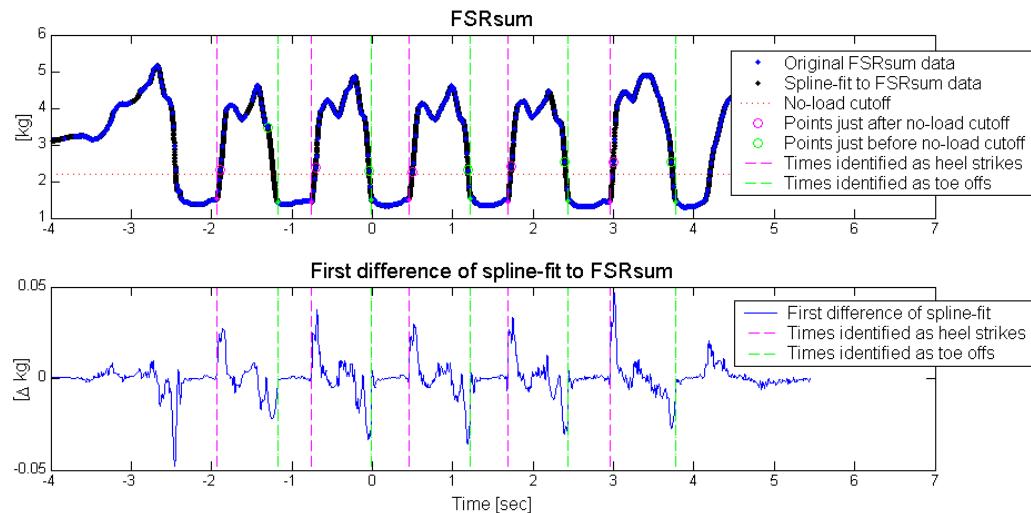


Figure 4.52 Determination of heel strike and toe off from FSRsum

1. See `hstos_fsrsum_spline.m`, an m-file available in Appendix F.2.

To locate heel strike and toe off, the first difference of the spline-fit, $D(FSRsum_{spline-fit})$ was calculated, as shown in Figure 4.52. The maximum peak of $D(FSRsum_{spline-fit})$ corresponded to the peak loading during heel strike, and was located using the vicinity determined by $FSRsum_{noload+}$. The heel strike time was set as the first spline-fit time point prior to the maximum peak where the condition $D(FSRsum_{spline-fit}) < \Delta 0.005$ kg was met. This condition was set after inspection of many gait trials; as shown in Figure 4.53, $\Delta 0.005$ kg was greater than the small changes of $FSRsum$ registered during swing, and indicated the start of the rapid increase in force corresponding to heel strike. This was non-objective method of finding the point corresponding to initial loading of the FSRs; for future work with the GaitShoe, the loading profile of the FSRs should be more fully investigated, perhaps by using a separate system where its time scale precisely aligned with the GaitShoe's time scale (as discussed in Appendix F.1, in the subject testing data, the time scale of the BML could only be aligned with the GaitShoe's time scale within 13.4 msec).

Similarly, the minimum value of $D(FSRsum_{spline-fit})$ corresponded to the peak unloading leading up to toe-off, and was located using the vicinity determined by $FSRsum_{noload-}$. The toe off time was estimated at the first spline-fit time point following the minimum where the magnitude of $D(FSRsum_{spline-fit})$ was less than $\Delta 0.005$ kg. As shown in Figure 4.53, $-\Delta 0.005$ kg approximately corresponded to the unloading of the $FSRsum$ due to toe-off and initiation of leg-swing.

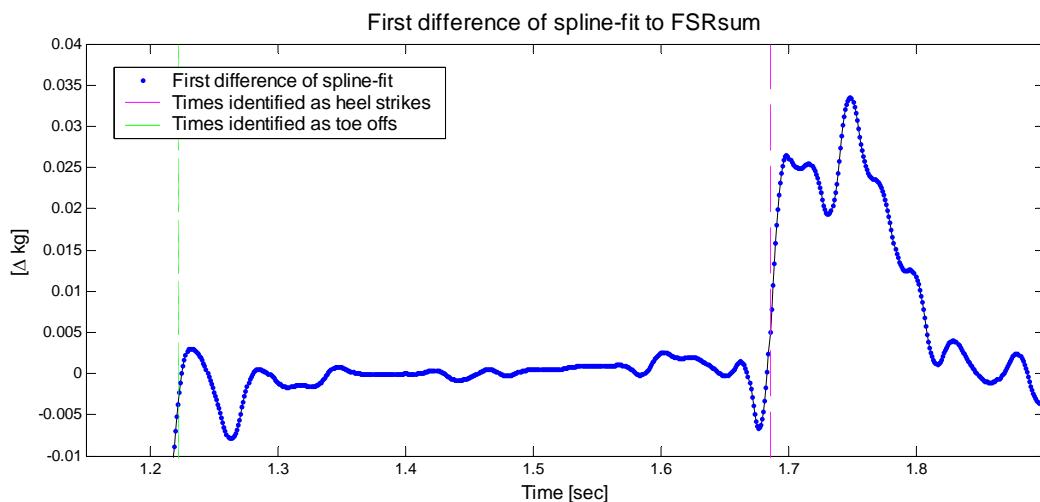


Figure 4.53 First difference of spline-fit, zoomed-in

Because there are two FSRs located under the heel, this method was expected to perform well in determining the heel strike. However, there are no FSRs located underneath the great toe, so this method was expected to estimate the toe off time early. Thus, these heel strike and toe off times were used to determine the integration bounds for the z-gyroscope, and the toe off time was subsequently re-evaluated.

Maximum Pitch Determination of Toe Off Timing

The final toe off times were determined by the location of the minimum pitch. As shown in Figure 4.16, the maximum pitch occurs at toe-off; as the foot rolls off the foot, the pitch increases, and once the toe is off the ground, a rapid acceleration at the start of leg-swing occurs, and results in a decrease in the pitch. Once evaluation of the z-gyroscope was complete, the times corresponding to the maximum pitch values during each stride were determined using the post-spline-fit method described in Section 4.3.2. These toe off times were compared to values for toe-off obtained from the BML analysis.

X-Accelerometer Upper Integration Bound for Alternate Heel Strike Timing

As described in Section 4.4.3, the upper integration bound for the x-accelerometer was set when the magnitude of $Ax_{dynamic}$ was either less than 0.2 m/s^2 , or when the value of $Ax_{dynamic}$ was greater than 0 m/s^2 . This second condition was expected to correspond to an especially strong heel strike. Each of the upper integration bounds were investigated, and if the bound corresponded to a positive spike in the $Ax_{dynamic}$ output, the time point was saved as an alternate measurement of heel strike timing. Where available, these alternate heel strike times were compared to values for heel strike obtained from the BML analysis, as were the heel strike times calculated from the spline-fit of the FSRsum.

Chapter 5

GAIT PARAMETER VALIDATION

The outputs of several sensors in the GaitShoe system were analyzed to generate clinically relevant gait parameters. The data were collected during subject testing (described in Appendix B), and was evaluated by comparison to data collected simultaneously by the Massachusetts General Hospital Biomotion Laboratory (BML). This chapter discusses the results of the comparison with BML data. The examples presented while describing the analysis model and the analysis techniques all use data from the same gait trial.

Testing of the fifteen subjects resulted in 270 total trials of gait. The GaitShoe pitch, stride length, and vertical displacement results for all of these trials were compared to the data collected simultaneously from the MGH Biomotion Lab (BML). The validation was carried out by comparing quantities of interest from each of these results: the maximum and minimum values of pitch and the times at which they occurred, the total stride length, and the maximum vertical displacement and the time at which it occurred. The magnitude differences and the percent differences, using the BML as the reference value where available, were determined.

The GaitShoe heel strike and toe off times were compared to heel strike and toe off times determined by a physical therapist by inspection of the BML force plate data. For each subject, approximately three heel strike times and three toe off times of each times were determined for each foot, for a total of 86 comparisons. The magnitude difference was calculated for the heel strike times and the toe off times.

5.1 Pitch

The maximum and minimum values and times of the GaitShoe pitch were determined¹ with a post integration spline-fit², as described in Section 4.3.2. The BML data were evaluated for outlying data points, which were removed, and the maximum and minimum values and times of the remaining BML data were determined. Figure 5.1 shows the GaitShoe pitch, BML pitch, and all extrema.

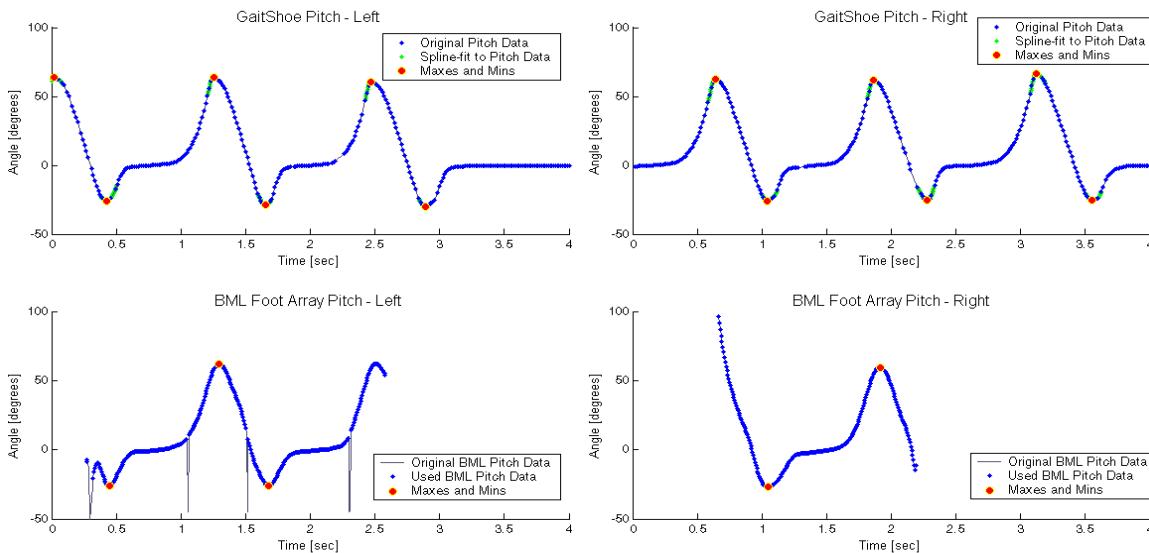


Figure 5.1 Comparison of GaitShoe pitch to BML foot array pitch

Figure 5.1 was included to demonstrate the issues which arise in the BML data. The lower left graph in Figure 5.1 shows that, like the GaitShoe data (see Section 4.1.3), there are occasional outliers in the BML data; these were excluded³ for the purposes of determining the maximum and minimum points of the BML foot pitch. In addition, there are often unusual data at the start and/or end of the BML data, usually due to only a portion of the array being visible to the cameras. These effects can be seen at the start of both BML foot pitch graphs in Figure 5.1: the lower left graph has an unusual dip before the true pitch

1. See `pitch_compare_postspline.m`, and `pitchpeakfinder.m`, m-files available in Appendix F.2.

2. See `gyroz_postspline.m`, an m-file available in Appendix F.2.

3. See `findmghoutliers.m`, an m-file available in Appendix F.2.

minimum at the start, and the lower right graph has an unusually large magnitude of pitch at the start; thus the first 16 and last 16 BML data points, corresponding to approximately 0.1 sec, were also excluded.

During determination of the extrema for both the GaitShoe data and the BML data, the number of dropped packets (in the BML data, dropped packets are the result of removal of outlying data points) in the vicinity of the extrema were stored, as well as the size of the largest number of successive dropped packets over the entire data series. Extrema were not included in the comparison if either the GaitShoe or the BML data had a series of dropped packets longer than 0.2 sec during the gait trial, or if either had 2 packets dropped in the vicinity of the extrema. This resulted in 279 comparisons of maximum pitch and time, and 241 comparisons of minimum pitch and time. The results are summarized in Figure 5.1, and histograms for each are shown in Figure 5.2. The percent change was not calculated for the time values, since the BML data, as seen in the lower right graph of Figure 5.1, did not always contain two successive pitch maximums or two successive pitch minimums from which to calculate a reference time.

TABLE 5.1 Pitch Validation Results

Comparison	Difference		Percent Change [%]		Samples
	Mean	Std Dev	Mean	Std Dev	
Shoe Pitch Max minus BML Pitch Max	1.9°	6.7°	4.22	10.4	279
Time of Shoe Pitch Max minus Time of BML Pitch Max	-.04 sec	.02 sec			279
Shoe Pitch Min minus BML Pitch Min	-4.9°	5.1°	29.1	20.5	241
Time of Shoe Pitch Min minus Time of BML Pitch Min	-.01sec	.02 sec			241

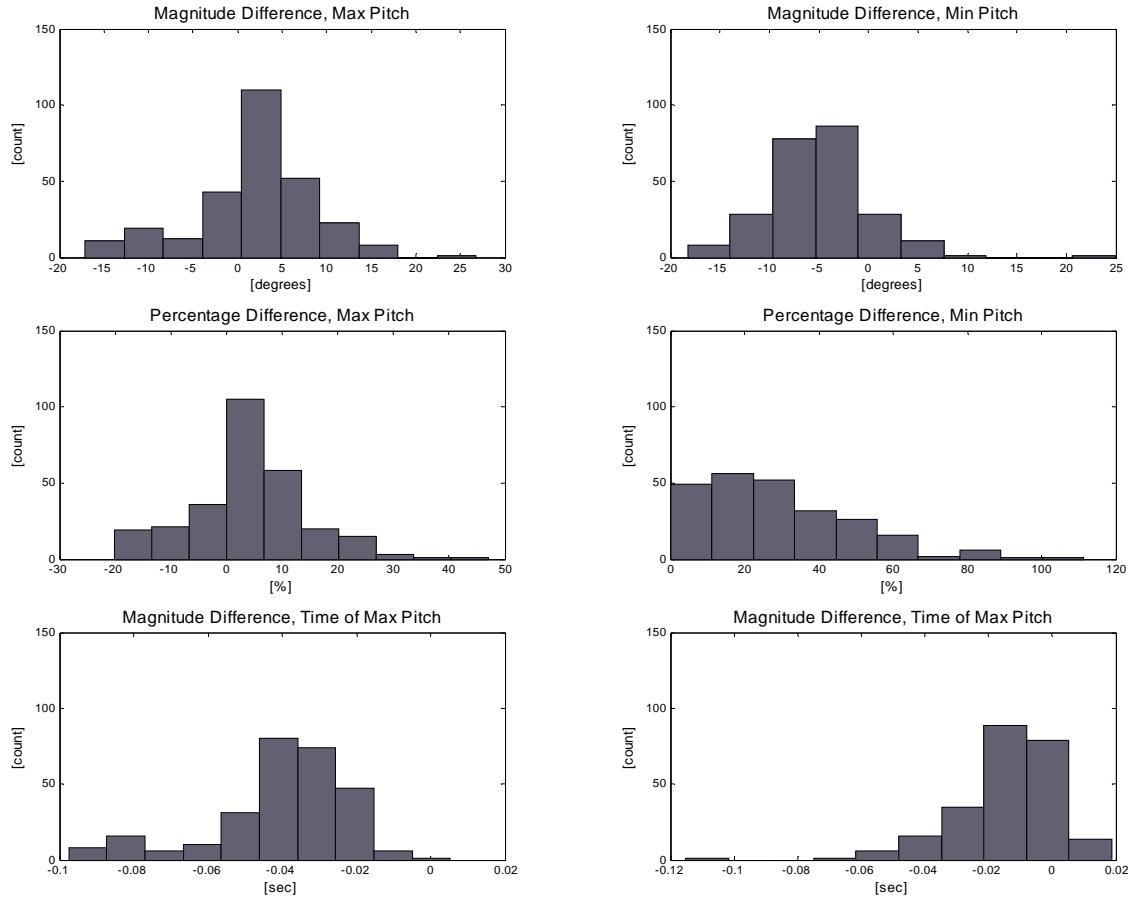


Figure 5.2 Histograms of the pitch validation results

5.2 Distance

The stride length (horizontal displacement) was determined by the GaitShoe via double-integration of the acceleration along the X_{room} axis, and, for comparison, along the X_{shoe} axis. The vertical displacement of the foot was determined via double-integration of the acceleration along the Y_{room} axis. These displacements were compared to the BML data samples if the BML data included the time corresponding to a lower integration bound and an upper integration bound (e.g. the BML data encompassed an entire swing phase); this resulted in 363 total samples available for comparison. For the stride length, the total displacements were compared, and for the vertical displacement, the peak displacements and the corresponding times were compared.

Comparisons were omitted if either the GaitShoe or the BML data had a series of dropped packets longer than 0.2 sec during the gait trial, or if either had a total of more than 10 packets dropped during the swing phase used in the comparison. This resulted in 166 comparisons of stride length, and 166 comparisons of peak vertical displacement and time. The results are summarized in Table 5.2; histograms for stride length are shown in Figure 5.3 and for peak vertical displacement and time in Figure 5.4 (the vertical axes are the same on all histogram plots; again, the percent change was not calculated for the time of peak vertical displacement, since an independent stride time was not always available from the BML data). Of note, while the magnitude of the mean difference of peak vertical displacement is only slightly larger than the mean difference of stride length, since the vertical displacement of the foot during stride is an order of magnitude smaller than the stride length (on the order of 10 cm as compared to more than a meter), the percentage change in peak vertical displacement is unacceptably large, at 72.7%. This is most likely mainly due to small errors in the pitch that have affected the determination of the acceleration in the Y_{room} axis.

TABLE 5.2 Displacement Validation Results

Comparison (All GaitShoe minus BML)	Difference		Percent Change		Samples
	Mean	Std Dev	Mean	Std Dev	
Stride Length <i>Double linear integration of Ax-room_{dynamic}</i>	6.1 cm	15.4 cm	5.4%	12.9%	166
Stride Length <i>Double linear integration of Ax-shoe_{dynamic}</i>	-4.0 cm	20.1 cm	-1.8%	17.3%	166
Peak Vertical Displacement	6.9 cm	3.3 cm	72.7%	45.8%	166
Time of Peak Vertical Displacement	-0.03 sec	0.06 sec			166

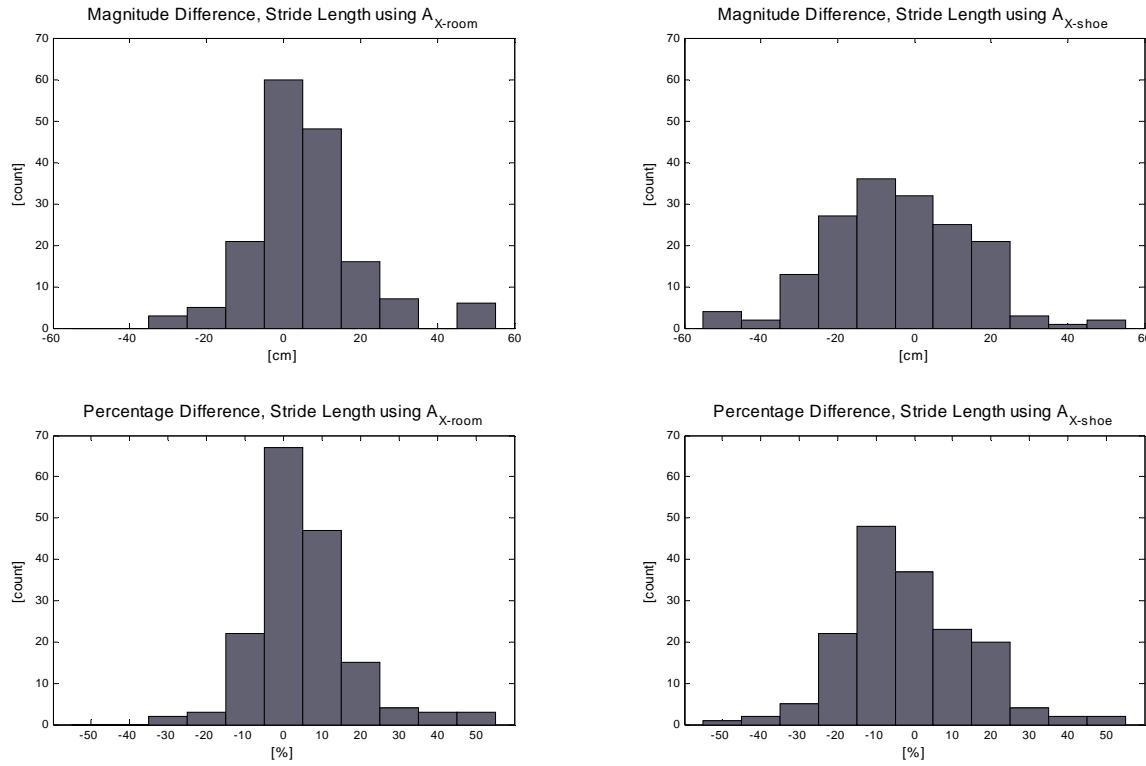


Figure 5.3 Histograms of the stride length validation results

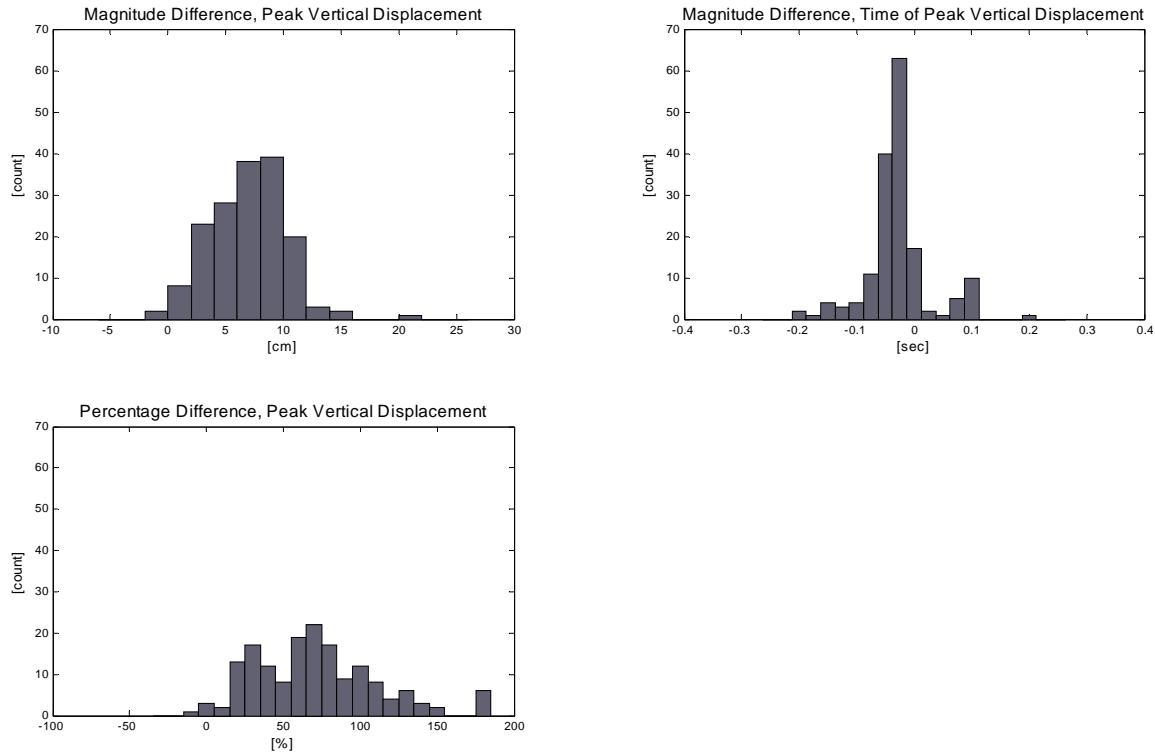


Figure 5.4 Histograms of the vertical displacement validation results

5.3 Heel Strike and Toe Off

The heel strike and toe off times determined by the GaitShoe were compared to times determined by a physical therapist at the MGH Biomotion Lab, by inspection of the force plate data. A total of 86 samples including both heel strike and toe off time were provided from the for validation; of these, nine were excluded, by inspection, because of obvious errors (such as a BML heel strike occurring when the GaitShoe FSRs did not have any load); these should be investigated further to find the reason for the errors. Two additional samples were excluded from toe off analysis because the IMU became loose towards the end of testing of one of the early subjects, and so the gyroscope output was not available.

As explained above, two methods of determining heel strike were evaluated. The first method used only the FSRsum to detect loading corresponding to heel strike; the second method used a spike in the x-accelerometer data corresponding to a strong floor impact where available and the FSRsum method if the x-accelerometer was not jolted. The toe off time was evaluated using the time of occurrence of the maximum pitch. The results are summarized in Figure 5.3, and histograms for each are shown in Figure 5.5.

TABLE 5.3 Heel Strike and Toe Off Validation Results

Comparison	Difference [msec]		Samples
	Mean	Std Dev	
Shoe Heel Strike minus BML Heel Strike <i>FSRsum method only</i>	-14.4	21.5	77
Shoe Heel Strike minus BML Heel Strike <i>FSRsum and x-accelerometer method</i>	-6.7	22.9	77
Shoe Toe Off minus Toe Off	-2.9	16.9	75

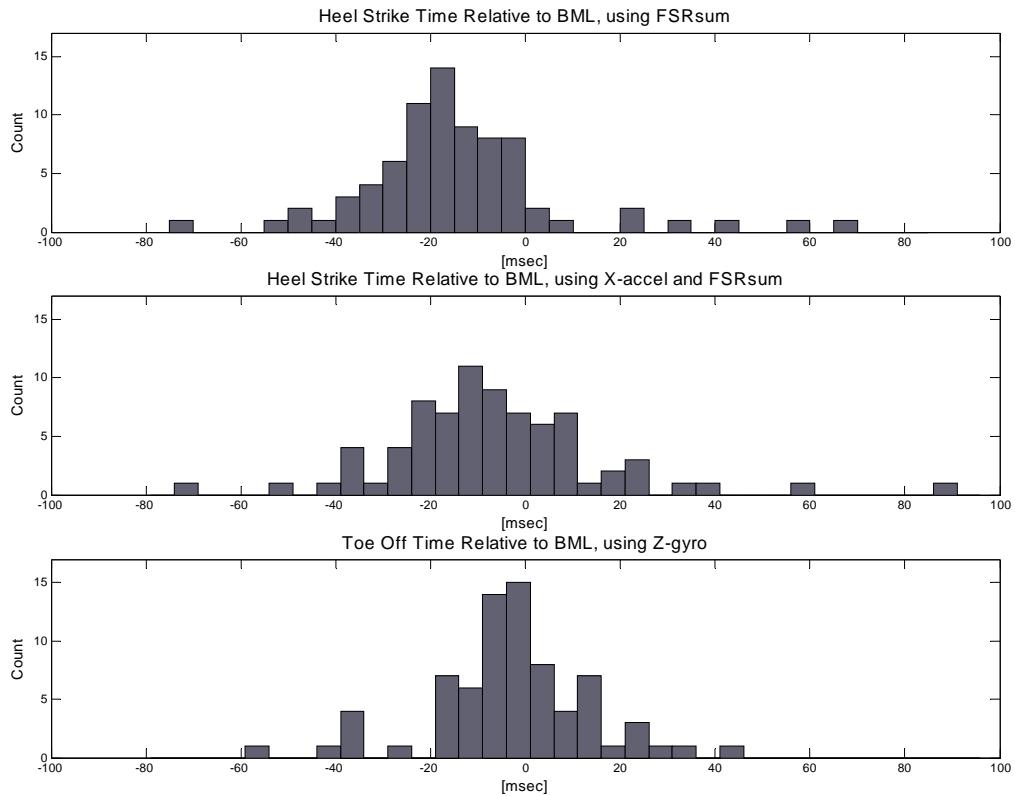


Figure 5.5 Histograms of the heel strike time and toe off time validation results

5.4 Discussion

The overall results from this initial analysis are encouraging, though a more detailed analysis may be needed to improve the results.

The GaitShoe's calculation of maximum pitch correlate well with the BML reference pitch, with a mean difference of 1.9° (standard deviation 6.7°), which corresponded to a mean percent change of 4.22% (standard deviation 10.4%). The difference in time between the maximum pitch points in each group was 40 msec. However, the calculation of minimum pitch did not correlate quite as well. The mean difference was -4.9° (standard deviation 5.1°), but because the magnitude of the minimum pitch is typically on the order of $10\text{--}15^\circ$, this corresponded to a mean percent change of 29.1% (standard deviation

20.5%). The minimum pitch suffers from cumulative integration errors, as it occurs toward the end of the stride.

The stride length results are very encouraging. Surprisingly, the mean difference is better when the stride length is determined via integration along the X_{shoe} axis than via integration along the X_{room} axis; however the standard deviation is much larger along the X_{shoe} axis. As compared to the BML stride length, the stride length calculated along the X_{shoe} axis has a mean difference of -4.0 cm (standard deviation 20.1 cm) and percent difference of -1.8% (standard deviation 17.3%), while as compared the stride length calculated along the X_{room} axis has a mean difference of 6.1 cm (standard deviation 15.4 cm) and percent difference of -5.4% (standard deviation 12.9%). These results are encouraging considering the large number of assumptions made in the analysis; by incorporating the full analysis of all three gyroscopes and all three accelerometers, both the mean difference, and the standard deviation are likely to decrease. The results of stride length calculated along the X_{shoe} axis suggests that if smaller and less expensive system were required for certain applications analyzing straight-line gait, a single-axis accelerometer could be used with a single gyroscope.

Though the difference for the vertical displacement as compared to the BML results is of similar magnitude to the stride length difference, the percent change is much larger. A typical stride length is longer than a meter, while vertical displacement is generally under 10 cm. Thus, while the vertical displacement as compared to the BML results has a mean difference of 6.8 cm (standard deviation 3.8 cm), the percent difference is -72.7% (standard deviation 50.0%). The time of the peak displacements as compared to the BML results has a mean difference of -.03 sec (standard deviation 0.06 sec). Unlike the stride length, these results appear to have suffered from the large number of assumptions made in the analysis, but will hopefully improve once the analysis is broadened to include all three gyroscopes and accelerometers.

In addition, errors in the pitch are likely to strongly contribute to the large errors in the vertical displacement, as well as the result that the stride length calculated along the X_{shoe} axis is as accurate as or better than the stride length calculated X_{room} . Determining the dynamic acceleration requires the orientation of the accelerometers with respect to the room, Φ_x and Φ_y , which correspond to the pitch. However, in addition, determination of the accelerations along the Y_{room} and X_{room} axes from the dynamic acceleration uses relationships involving the sines and cosines of Φ_x and Φ_y . Thus, errors in the pitch affect the calculations along the Y_{room} and X_{room} axes more than the calculation along the X_{shoe} axis. Improvements in the pitch calculation are therefore likely to improve the results of the stride length and vertical displacement along the Y_{room} and X_{room} axes. In particular, a hardware change to use the Analog Devices gyroscope to measure rotation about the z-axis, rather than the Murata gyroscope, may sufficiently improve the pitch calculation; as discussed in Section 3.3.2, the Analog Devices gyroscope has two vibrating structures operating in anti-phase to reduce common mode signals unrelated to angular velocity (such as external shocks or vibrations).

The results for the heel strike time and toe off time are particularly interesting. Both methods of calculating the heel strike time anticipate the time determined by the BML; the method using the FSRsum has a mean difference of -14.4 msec (standard deviation 21.5 msec) as compared to the BML time, while the method using the x-accelerometer in combination with the FSRsum has a mean difference of only -6.7 msec (standard deviation 22.9 msec). These results suggest that it is possible that the automated analysis of the GaitShoe results in detection of heel strike earlier than the human interpretation of the force plate data. However, the time scales of the GaitShoe system and the BML system are only aligned within 13.4 msec (see Appendix F.1), such that the GaitShoe time may register up to 13.4 msec ahead of the BML time; this may account for the shift between the detection on the two systems. Further evaluation, perhaps with a different type of validation equipment, and certainly with an improved method of aligning the time scales, is recommended to confirm these results.

The toe off times are the most well correlated parameter in comparison with the BML system, with a mean difference of only -2.9 msec (standard deviation 16.9 msec). However, these results are moderately surprising, given that the maximum pitch was used to detect toe off (as the maximum pitch occurs right around the time of toe off), and the pitch results demonstrated that the time of detection of maximum pitch by the GaitShoe had a mean shift of 40 msec from the BML maximum pitch. It would be interesting to add FSRs underneath the great toe to see how the toe off time detected by an FSRsum including more FSRs compared to the BML and maximum pitch generated toe off times.

Chapter 6

PATTERN RECOGNITION ANALYSIS

The ability of the GaitShoe system to provide information capable of identifying gait patterns was investigated by using standard pattern recognition techniques with the data collected during subject testing. Two classification problems were investigated; the primary goal was to distinguish the gait of subjects with Parkinson's disease from the gait of subjects with normal gait. Of secondary interest was recognizing individual subjects from their gait. This chapter discusses the techniques used, the selection of the subject data that was classified, the selection of the feature set, hypotheses, training and testing data, and the results of the classification.

6.1 Pattern Recognition Techniques

Several classic techniques were applied to data sets extracted from the subject testing, in order to investigate which technique was best at classifying this type of data. These techniques are described briefly here; all employ supervised learning, where a labeled training set was used to train the classifier, and a separate labeled test set (containing samples not in the training set) was used to evaluate the classifier.

6.1.1 Classification and Regression Trees (CART)

CART is a decision tree tool, distributed by Salford Systems software. It was developed from original work by Breiman, Friedman, Olshen, Stone at the Stanford University and

the University of California at Berkeley [93]. It uses binary recursive partitioning; each node is split into exactly two nodes, and the process is repeated with each child node. The CART software analyzes the data through a set of rules which determine the following: node splitting (to create the tree), tree completion, tree pruning, terminal node classification.

To determine the split of each node, the CART conducts a brute force search by calculating all possible splits for all features. For continuous data (as opposed to categorical data), such as was generated from the GaitShoe data, a split is a numerical point which forms all or part of the boundary that separates two classes. For example, in gait data there might be a split in stride length at 1.4 m, such that stride lengths greater than 1.4 m correspond to male gait, while stride lengths shorter than 1.4 m correspond to female gait. The brute force search finds the split for each of the features for all samples in the data set, and then selects the best split for each node using a set rule. The Gini rule was the default in CART; the implementation of the Gini rule in CART was set to separate classes of data one at a time, by trying to isolate the classes in order of importance (most to least) [94]. By default in CART, importance is defined by the number of samples in the class, with larger classes more important (classes can be weighted; this option was not used). Thus, the split that provides the highest separation of the largest class was selected.

The splitting process is repeated on each resulting node until the tree is completed, as determined by the default settings: splitting was stopped when either fewer than ten samples were in a node or when all samples in the node were of a single class.

Once the tree is completed using the training set, CART uses the testing set of data to prune the split nodes to select the "best" tree, determined by the default "standard error rule," which picks the tree with the minimum cost. The cost is calculated by the rate at which the samples in the test set are misclassified, and is determined for both the full tree and for all possible sub-trees, with a penalty for large trees (because the testing set is used to select the final tree, this results in an overly optimistic estimation of error rates). This

method is "overly optimistic" in that the final decision for the tree is determined by the testing set, rather than determined independently of the testing set.

One of the great benefits of CART is that the tree provides information about the most useful parameters in the feature set, as the splits at each of the nodes represent the features which best classify the data. In addition, the data set can have any number of classes, and the software completes the classification in seconds. There were two limitations to the software: the interface is not readily automatable, making analysis of several data sets tedious, and CART does not directly report or store which data were misclassified [94].

6.1.2 Bayes Decision Theory & Naïve Bayes

Bayes decision theory is a classification method based on probabilities. It assumes that the classification problem can be described in probabilistic terms, and that all the underlying probability values are known or can be reasonably approximated.

In order to calculate the posterior probability, $P(\omega_j|x)$, that x is in class ω_j , the following probability values¹ must be obtained or estimated: 1) the state-conditional probability density function, $p(x|\omega_j)$, which is the likelihood that x is observed, given class ω_j ; and, 2) the prior probability, $P(\omega_j)$, of class ω_j , which is the probability that any observation is class ω_j (all prior probabilities sum to one). Bayes rule, shown in Eq. 6.1, states that the posterior probability that sample x is in class ω_j is equal to the likelihood of x multiplied by the prior probability of class ω_j and divided by the evidence, $p(x)$, defined in Eq. 6.2. The evidence is a scale factor to guarantee that the posterior probabilities over all classes sum to one. Bayes rule was published posthumously, in 1763, by mathematician Rev. Thomas Bayes [95].

$$P(\omega_j|x) = \frac{p(x|\omega_j)P(\omega_j)}{p(x)} \quad (6.1)$$

1. P represents a probability, and p represents a probability density

$$p(x) = \sum_{j=1}^n p(x|\omega_j)P(\omega_j) \quad (6.2)$$

For example, Bayes rule would state that the probability that a subject with a stride length of 1.3 m is a female is equal to the likelihood that female subjects have a stride length of 1.3 m multiplied by the probability that the subject is female, and divided by the evidence. The evidence would be equal to the sum of the numerator plus the likelihood that male subjects have a stride length of 1.3 m multiplied by the probability that the subject is male.

For the two class problem, both posterior probabilities, $P(\omega_1/x)$ and $P(\omega_2/x)$, are calculated and x is classified as class ω_1 if $P(\omega_1/x) > P(\omega_2/x)$; otherwise, x is classified as class ω_2 , where the data sample is either a single feature x , or a vector \mathbf{x} of several features.

When \mathbf{x} is a vector of several features, information about the interdependence of the features is required to estimate the prior probabilities. Naïve Bayes is a technique which simplifies the estimation of the prior probabilities by assuming that class-conditional independence exists. This assumes that all features are independent from each other, so the prior probability can be simply calculated from the prior probabilities of each individual feature, as shown in Eq. 6.3.

$$p(\omega_j|\mathbf{x}) \propto \prod_{i=1}^d p(x_i|\omega_j) \quad (6.3)$$

Though the class-conditional independence assumption rarely holds in practice, Naïve Bayes classifiers are easy to implement, and often perform reasonably well. In addition, multiple classes can be considered as easily as two classes [96].

Use of Bayesian approach is usually to update the prior probabilities as new information becomes known; in this case, the analysis is carried out once. Since the Naïve Bayes assumptions are unlikely to apply to the gait feature data, which are certainly not indepen-

dent, this was not expected to provide the best classification, but was included to see how well a simple technique could perform.

For this work, an implementation of Naïve Bayes in a program called Weka was used. Weka is a collection of machine learning algorithms for data mining tasks, written in open-source Java, and developed at the University of Waikato in New Zealand. The probabilities were estimated using the (default) normal distribution, and the classification was complete in a few seconds [97].

6.1.3 Support Vector Machines (SVM)

SVMs are a classification method based on a linear learning machine (a learning algorithm that uses linear combinations of the input variables) and were first introduced in 1992 by Vapnik and co-workers, at the Computational Learning Theory conference [98].

SVMs use linear discriminant functions, which means that the form of the underlying function is known (or assumed), and the training data are used to estimate the values of the parameters of the classifier. Unlike Bayes methods, SVMs require no knowledge or assumptions of the underlying probability distribution of the samples.

The essential function of SVMs is to find the optimal hyperplane that separates the labeled samples into two categories. The optimal hyperplane is defined to be the one that has a maximal distance to the closest training samples (see Eq. 6.4, below). This distance is called the margin, and the closest training samples are called the support vectors. The support vectors are the most informative samples; the hyperplane could be recreated from their information alone.

A larger margin corresponds to a SVM classifier with better generalization properties, in the sense that the distance between the hyperplane and the support vectors corresponds to the separation between the two data classes. The complexity of a SVM classifier is indicated by the number of support vectors; more support vectors indicate a more complex classification.

Training data are not expected to be linearly separable in the feature space. Thus SVMs use a "kernel" to map the feature to a higher dimensional space where separability is more likely. In this case, it is useful to replace the dot product with a kernel function, to map the data to a higher dimensional space where it may be linearly separable by a hyperplane. Many functions can be used as a kernel function. While use of kernels can be a powerful technique for separating data, use of kernels can result in a feature space that overfits the training data, but does not generalize well to new data despite separating the training data well.

In practice, the generalized optimal hyperplane is found by maximizing the margin and minimizing the classification error in this new space. For example, let x_i be the original features, then given a hyperplane defined by $w_1x_1 + w_2x_2 + \dots + w_Nx_N = b$, the generalized optimal hyperplane is found by minimizing

$$\Phi(w, \xi) = \frac{1}{2}\|w\|^2 + C \cdot \sum_i \xi_i, \quad (6.4)$$

where ξ_i is some error function, and C is a weighting factor to weight the importance of maximizing the margin vs. minimizing classification error. By using Lagrange undetermined multipliers, and the Kuhn-Tucker construction, this optimization can be carried out by maximizing

$$L(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{k,j}^N \alpha_j \alpha_k z_j z_k K(x_1, x_2), \quad (6.5)$$

subject to the constraints

$$\sum_{k=1}^N z_k \alpha_k = 0 \quad \alpha_k \geq 0, k = 1, \dots, n, \quad (6.6)$$

given the training data. $K(x_1, x_2)$ refers to the kernel used, and the α_k are the undetermined Lagrange multipliers, in \mathbf{z} space.

Many implementations of SVMs are available [99]; the Matlab implementation by Steve Gunn was used [100] [101]. This classifier was easily automated within Matlab, and in addition, the classifier ran quickly, generally taking under 20 seconds. The data were treated as linearly separable in the native feature space (the dot product was not replaced with a specialized kernel).

SVMs do not provide information about which features are most informative. In addition, in most implementations, including the one used, SVMs can only classify two categories (it is possible to use SVMs to classify more than two categories, by serial analyses, but this is considerably more complex, and was not used) [96] [101] [102].

6.1.4 Neural Networks

An artificial neural network is an adaptive learning method that represents the data through the use of a parallel network of nodes ("neurons") arranged in layers and connected by weighted links. The link weights are determined by the training data.

A neuron with a single input is shown in Figure 6.1, left. The neuron is simply a model for processing the scalar input p via multiplication by weight w and addition of bias b . This sum, net input n , is the input to function f , which determines the output a . The function f can take any form, from a hard limit function, where $a = 1$ if $n \geq 1$ and $a = 0$ if $n < 0$, to a linear function, where $a \propto n$, to a sigmoid function, such as $a = 1/(1+e^{-n})$. This particular sigmoid function results in an output a that has a range of 0 to 1, for any value of n between plus and minus infinity. Sigmoid functions are often referred to as "squashing" functions because they compress the input into a predefined range; a key feature of all sigmoid functions is that they are differentiable.

The neural network is built by composing layers consisting of multiple neurons, where the input is usually a vector of multiple features. Figure 6.1, right, shows a vector input \mathbf{p} and a single layer; the weights form a matrix \mathbf{w} and the biases form vector \mathbf{b} . Neural networks are not limited to a single layer: multiple layers can be added serially, by using the outputs

of a the first layer as inputs to the next layer. The final layer is called the output layer, and all other layers are called hidden layers. For pattern recognition, the neural network is set up such that the final layer has as many neurons, and thus as many outputs, as there are classes.

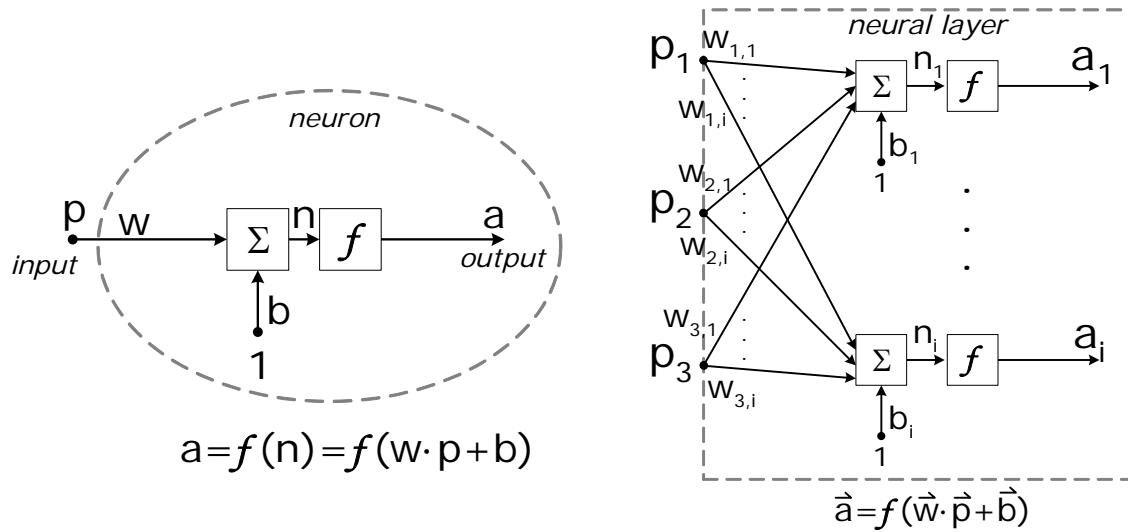


Figure 6.1 Examples of a single input neuron and a multiple input neural layer

Backpropagation is a frequently used method to improve the accuracy of neural network. In backpropagation, an effective error is calculated for every neuron, and this error is used to adjust the weights and biases; the sigmoid function is commonly used in backpropagation. To train a network, the training data is first applied to the network, in order to generate the output vectors, \mathbf{a} . Next, the error is calculated, by comparing \mathbf{a} to the labels on the training data; generally, a sum-of-squares error or mean-squared error is used, such that the error is proportional to $\sum_{s=1}^S \sum_{i=1}^I (a_{i,s} - b_{i,s})^2$, where I is the total number of neurons, and S is the total number of samples in the training data. This type of error is useful analytically, because the errors for each output $a_{i,s}$ remain independent, and are simply summed together to calculate the overall error.

There are many variations of the error adjustment algorithm used in backpropagation. The basic implementation adjusts the weights and biases in the direction of the negative of the

gradient, which corresponds to the direction in which performance degrades most quickly, such as

$$\hat{x}_{k+1} = \hat{x}_k - \alpha \hat{g}_k, \quad (6.7)$$

where \hat{x}_k is a vector containing the current weights and biases, \hat{g}_k is the current gradient, and α is the learning rate. The learning rate is a parameter that can be set to control how much the weights and biases are updated, where a number closer to zero corresponds to a slower learning rate; typical values are between 0.05 and 0.75 (α is always positive).

There are two approaches for updating the weights and biases: batch learning and on-line learning. In batch learning, \hat{g}_k is calculated from the sum of the individual gradients of the errors for all samples in the training set, given \hat{x}_k , and the weights and biases are updated across all samples. Here, the subscript k refers to a single pass through all the training set; such a pass is called an epoch. Batch learning updates the weights and biases slowly, so many epochs are usually required to minimize the error. However, \hat{g}_k is a good approximation of the true gradient.

In on-line learning, a sample is selected from the training set, the gradient \hat{g}_k is calculated for that sample, and the weights and biases are immediately updated. Here, the subscript k refers to a sample. In a single epoch, on-line learning updates the weights and biases as many times as there are samples, whereas batch learning updates only once (typically, the samples are selected in a random order each epoch, so as to avoid cyclic effects). However, the individual \hat{g}_k gradients are essentially noisy estimates of the true gradient, so this approach no longer approximates the true gradient. Thus, at individual samples, the updates may result in a larger overall error. However, although the path may not be direct, overall, the error will decrease; however, the randomness in on-line learning results in a jitter about the minimum. This generally prevents the network from getting stuck at a local minimum, which can happen in batch learning, because the jitter allows the on-line learning algorithm a method to move out of a local minimum.

Another parameter which is often incorporated into backpropagation is a parameter called momentum, μ . The purpose of momentum is to prevent a network from being stuck in a local minimum, by adjusting the weights not only in response to the local gradient, but also in response to recent trends in the error. It can be incorporated by including a fraction of the previous weight change, $\Delta\hat{x}_{k-1}$, such as

$$\hat{x}_{k+1} = \hat{x}_k - \alpha\hat{g}_k + \mu\Delta\hat{x}_{k-1}, \quad (6.8)$$

where the momentum, μ , is between 0 and 1 [96][103][104].

For this work, the implementation in Weka was used (Matlab 6.5 also contains an extensive neural network toolbox). The Weka neural network uses backpropagation to train, with sigmoid functions at the nodes. The default values for the training parameters were used: the learning rate was 0.3, the momentum was 0.2, the number of hidden layers was set to one half the sum of the number of features plus the number of classes, and the number of epochs was 500. There was also a validation threshold, set to 20, which limited the number of times in a row the error could get worse before training was stopped, and a reset flag, set to true, which allowed the weights and biases to be reset and the training restarted with a lower learning rate, if the network diverged from the training output. The final values of the weights and biases were determined with the training data, and subsequently evaluated by the testing data. This implementation completed the training in under a minute [97].

6.2 Data Sets and Classifications

Since the primary goal was to investigate the use of pattern recognition techniques to distinguish Parkinsonian gait from normal gait, using data from the GaitShoe, it was important to select the data sets carefully so as not to bias the results. Factors which contribute to natural variations in gait were examined, and the subject data available was evaluated for occurrence of these factors. The data classes were selected in an effort to minimize differences between classes due to known causes of variations in gait, as discussed below.

6.2.1 Gait Variation

Several factors which contribute to variation in gait were considered. Over a general population of people under age 65, there is a standard deviation of 10% across gait parameters such as velocity or stride length. This variation across the population is due to a number of factors, such as gender, age, and subject height [105].

Gender

There are statistical differences between the gait of males and of females. These are summarized in Table 6.1. In general, males have a longer stride length, but take fewer steps per minute, as compared to females. Overall, these two factors combine to give males a slightly faster velocity than females [105].

TABLE 6.1 Gender differences in gait [105]

	Males	Females
Mean stride length [m]	1.46	1.28
Mean cadence [steps/min]	111	117
Mean velocity [m/min]	82	77

Age

In healthy subjects with normal gait¹, age does not have a measurable effect on gait until the population includes subjects older than 60 years. For a group of subjects aged 60 to 65 years, the mean velocity was found to decrease by 3%, while for a group aged 60 to 80 years, the mean velocity decreased by 9% [105].

Leg Length

In adults, there is a weak correlation between stride length and leg length during walking gait (a stronger correlation exists during running gait). A study which grouped 120 male

1. When arthritis and other disabilities which affect gait are included, studies of gait in older adults have shown a 14% deviation in velocity [105].

adults into equal groups of short, medium, and tall heights (leg length generally corresponds to height) found only a 4% change between the mean stride lengths in each group [105].

6.2.2 Subjects and Data Sets

The fifteen subjects (detailed subject information is available in Appendix B.2) were divided into five subgroups, summarized in Table 6.2 by presence of PD, gender, and age. As discussed above, gender and age may affect parameters of gait such as stride length and velocity. With large numbers of subjects, the number of males and females would likely be balanced, however, in this small study, only three out of the ten subjects with normal gait were male. This resulted in a concern that differences between male and female gait might affect the classification.

TABLE 6.2 Summary of subject groupings

Group	Parkinson's disease	Gender	Lowest Age [years]	Highest Age [years]	Number of subjects
1	No	Female	24	28	5
2	No	Male	25	30	3
3	No	Female	48	54	2
4	Yes	Female	53	65	3
5	Yes	Male	64	76	2

In addition, the subjects with PD all had an age greater than 50 years; in particular, the male subjects with PD were 64 and 76, while the male subjects with normal gait were all 30 years old or younger. However, the seven females with normal gait had two clusters of ages: five subjects in their mid-twenties, and two 48 and 54 years old. The three females with Parkinson's disease were in their fifties and sixties, so the female subjects with normal gait had two subjects who could be considered age-matched to the female subjects with PD. Thus, to not bias the classification results, the groups selected for data sets included only the ten female subjects.

In regards to the effect of leg length¹, the ten females had a mean height of 1.62 m, and a standard deviation of 0.03 m (the tallest subject was 1.68 m, and the shortest was 1.57 m). It was expected that, given the small spread of heights across the ten females, leg length was unlikely to contribute significantly to any changes between the groups.

The groups selected for the classification are summarized in Table 6.3. Class "NLFY" ("young" females with normal gait) had a total of 92 trials across five subjects, class "NLF" (age-matched females with normal gait) had a total of 37 trials across two subjects, and class "PDF" (age-matched females with Parkinson's disease) had a total of 64 trials. This sums to a total of 193 trials across all ten subjects.

TABLE 6.3 Classifications of the ten female subjects

Moniker	Group	Number of Subjects	Total Trials
NLFY	1	5	92
NLF	2	2	37
PDF	4	3	64

6.3 Feature Set

A key decision when building a classifier is the selection of features (the techniques described above all classify samples via a vector of single features about the sample; none evaluate the sample based on a data-time series). Clearly, the success of the classifier will greatly depend on whether the features encapsulate the distinguishing characteristics of the data classes.

The GaitShoe provides such a vast amount of information that selection of features can be viewed as an ongoing problem, with the features used here as an initial solution. The feature set used is detailed in Table 6.4. The feature set was developed to include a variety of

1. Though leg length was recorded via the BML calibration routines, body height was directly accessible from the subject information and was used for this comparison.

information about the gait, but without unduly influencing the outcome of the classification. Parameters traditionally expected to change with age, such as stride length and velocity were not included; however, the training process of a classifier should eliminate dependence on parameters that are not related to age, if age is not the classification goal, so parameters, such as foot pitch, that may be correlated with stride length and velocity were included.

Table 6.4 has five columns: the first numbers the features, the second lists the data type, the third explains the type of analysis used to extract the feature from the data, the fourth lists the metric used, and the fifth is a description of what the feature represents. Three types of analyses were applied: the mean of the data, the standard deviation of the data ("std dev"), and the clipped standard deviation (the standard deviation of the data with the top and bottom 10% of data removed; "std dev (clipped)").

Two metrics were used, "L and R combined" and "L to R ratio." The metric "L and R combined" was a combination of the data from the left and right feet, where the analysis was applied to a vector consisting of a concatenation of all the data from the left foot and all the data from the right foot (e.g. the right foot data vector was appended to the left foot data vector). The purpose of this metric was to extract an overall feature from both feet. Initial work included the left foot and the right foot data as separate features, but it was found that left foot and right foot features were often interchangeable, so they were combined for the final feature set.

The metric "L to R ratio" was the absolute value of minus one plus the ratio of the analysis applied to the data from the left foot divided by the analysis applied to the data from the right foot. The purpose of this metric was to get a measure of any asymmetry between the two feet (healthy gait is expected to be symmetric). One was subtracted from the ratio, and the absolute value was applied, so as to lump all the asymmetries together (this resulted in a slight skewing of asymmetries in which the value for the right foot is greater than the value for the left foot, but was not expected to affect the results significantly).

TABLE 6.4 Feature set

	Data Type	Analysis	Metric	Description
1	FSRsum/bodyweight	Std dev (clipped)	L and R combined	Walking energy variation
2	FSRsum/bodyweight	Std dev (clipped)	L to R ratio	Walking energy asymmetry
3	StepF/bodyweight	Mean	L and R combined	Step energy amplitude
4	StepF/bodyweight	Mean	L to R ratio	Step energy asymmetry
5	StepF/bodyweight	Std dev	L and R combined	Step energy variation
6	Gyro-x	Std dev (clipped)	L and R combined	Yaw variation
7	Gyro-x	Std dev (clipped)	L to R ratio	Yaw asymmetry
8	Gyro-y	Std dev (clipped)	L and R combined	Roll variation
9	Gyro-y	Std dev (clipped)	L to R ratio	Roll asymmetry
10	Gyro-z	Std dev (clipped)	L and R combined	Pitch variation
11	Gyro-z	Std dev (clipped)	L to R ratio	Pitch asymmetry
12	Accel-x	Std dev (clipped)	L and R combined	Forward motion variation
13	Accel-x	Std dev (clipped)	L to R ratio	Forward motion asymmetry
14	Accel-y	Std dev (clipped)	L and R combined	Upward motion variation
15	Accel-y	Std dev (clipped)	L to R ratio	Upward motion asymmetry
16	Accel-z	Std dev (clipped)	L and R combined	Sideways motion variation
17	Accel-z	Std dev (clipped)	L to R ratio	Sideways motion asymmetry
18	<i>Maximum pitch^a</i>	<i>Mean</i>	<i>L and R combined</i>	<i>Shuffle index</i>
19	Maximum pitch	Mean	L to R ratio	Asymmetric shuffle
20	Maximum pitch	Std dev	L and R combined	Shuffle variation
21	Minimum pitch	Mean	L and R combined	Shuffle index
22	Minimum pitch	Mean	L to R ratio	Asymmetric shuffle
23	Minimum pitch	Std dev	L and R combined	Shuffle variation
24	Percent stance time	Mean	L and R combined	Shuffle duration
25	Percent stance time	Mean	L to R ratio	Shuffle duration asymmetry
26	Percent stance time	Std dev	L and R combined	Shuffle duration variation

a. This parameter was ultimately excluded, as it possibly was age-dependent in this sample.

Features 1-5 involve the summed output from the FSR sensors. Features 1 and 2 involve the sum of the FSRs ("FSRsum") divided by the subject's body weight, and features 3, 4, and 5 involve the integration of the sum of the FSRs between heel strike and toe off ("StepF") divided by the subject's body weight.

Features 6 through 17 are the clipped standard deviations of the calibrated outputs of the three gyroscopes and the three accelerometers, with both metrics applied. The bend sensor outputs and the pvdf strip outputs were not used at all for the feature set, because of the likelihood that there was intra-subject variation due to the fitting of these sensors, as discussed in Section 4.6 and Section 4.7 respectively, rather than resulting from changes in the gait of the subject.

The remaining features, like features 3-5, involved quantities that were derived from sensor outputs during the gait parameter analysis. Features 18 through 23 involve the pitch of the foot, which comes from the integration of one of the gyroscopes, and as discussed in Section 4.3.2, the characteristic pitch of the foot has a maxima and a minima during each stride (see Figure 4.16 on p. 120). Features 18, 19 and 20 use the maximum pitch values, and features 21, 22 and 23 use the minimum pitch.

However, feature 18, the mean of the maximum pitches in the left and right feet, was excluded when it was seen that it may have had a correlation with age in this sample. Initial work with CART revealed that when the older of the two NLF subjects was used as the test set, feature 18 was used as a splitting criteria, and the test set was grouped with the PDF group.

This feature, and feature 20, the mean of the minimum pitches in the left and right feet, are shown for the data samples of each subject, plotted against the subjects' ages in Figure 6.2. While it is impossible to say conclusively whether either of these features have some age-dependency, the mean maximum pitch does seem to be significantly less for the subjects older than 50 years, while the minimum pitch has a lesser degree of separation by age. Though it generally trends upwards, the mean minimum pitch of the subjects older

than 50 overlaps with those younger than 50 by more than 10 degrees out of a full range of 30 degrees (approximately 30% overlap), while the mean maximum pitch of the two groups overlap by under 5 degrees out of a full range of 40 degrees (less than 13% overlap). While the training of the pattern recognition system should exclude age-related parameters if not classifying by age, it was decided to exclude the mean maximum pitch as a feature (both of these parameters will be interesting to investigate further, with larger subject groups).

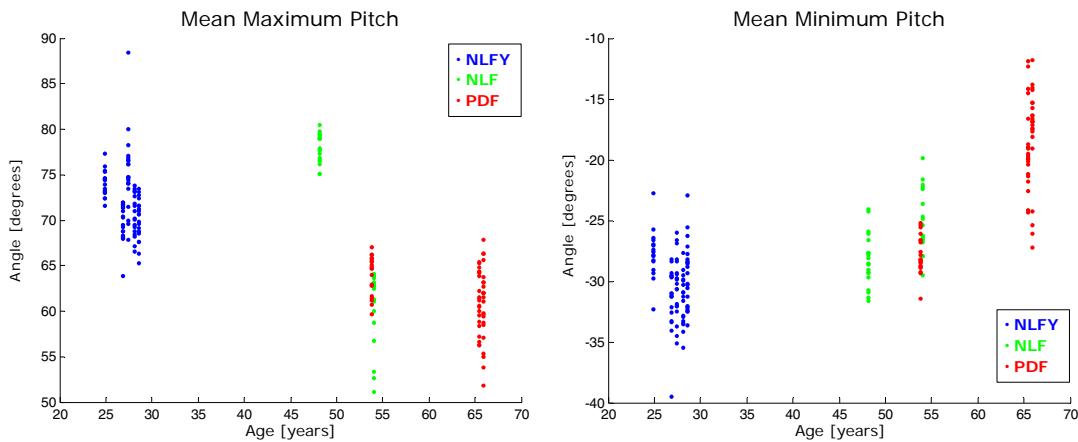


Figure 6.2 Plots of maximum and minimum pitch, by age

The final features, 24 through 26, use the calculations of heel strike and toe off timing to determine the percentage of the step each foot spends in stance ("percent stance time").

In feature generation from the data which were not continuous time series, there were often only a few measurements available in each sample, for a few reasons. There were usually five to seven footsteps per sample, so the number of total measurements was small. The integrated FSR sum and the pitch extrema measurements were likely to be adversely affected by dropped packets, so these data were analyzed for dropped packets in the region of each of the individual measurements for that sample. If there were a significant number of dropped packets, as listed in Table 6.5, the measurement was not used in determining the feature. In evaluating the percent stance time, which was calculated by

$$\text{Percent Stance Time} = \frac{T_{TO_k} - T_{HS_k}}{T_{HS_{k+1}} - T_{HS_k}}, \quad (6.9)$$

the last step was not used, in order to exclude any variance due to slowing at the end of the gait trial. This meant that samples that had only four footsteps total only had two usable measurements of percent stance time, since calculation of percent stance time required two successive heel strikes.

TABLE 6.5 Measurement exclusion due to dropped data

Data	Exclusion Criteria
Integrated FSR sum	10 dropped packets between heel strike and toe off
Maximum and minimum pitch	5 dropped packets in vicinity of each extrema

Because of all these factors, a few of the samples have only one or two measurements for either the left foot or the right foot or both, which meant the standard deviation could not be calculated for that foot. Thus, the ratio of the standard deviation was not used as a feature for the analysis of these data; there were no samples that had only two or fewer measurements total for the left and right feet together, so the standard deviation of the combined left and right feet measurements was used.

Again, this feature set of 25 features is an initial attempt to characterize the gait. There may be other features which are more informative.

6.4 Hypotheses

The subjects used during gait testing for validation of the sensor outputs provided the opportunity to investigate changes in gait of subjects with Parkinson's disease, as compared to gait of normal subjects.

Three hypotheses were investigated. The first hypothesis was that the NLFY and NLF groups were likely to have a high degree of confusion, because the only known difference between the two groups was the age difference. For the three techniques that could handle more than two classes (CART, Naïve Bayes, and Neural Nets), Hypothesis 1 was tested with the three class problem of NLFY, NLF, and PDF. Since the SVM implementation

used could only handle two classes, three two class problems were tested: Hypothesis 1.1, NLFY and NLF; Hypothesis 1.2, NLF and PDF; and, Hypothesis 1.3, NLFY and PDF.

Assuming that Hypothesis 1 would be supported, the second hypothesis was that the PDF group would be highly separable from NLFY and NLF grouped together. Hypothesis 2 is a two class problem, NLFY/NLF and PDF, and was tested using all four techniques.

The third hypothesis was that the feature set was likely to contain enough information to classify individual subjects. Hypothesis 3 had ten classes, corresponding to each of the ten subjects, and was tested using CART, Naïve Bayes and Neural Nets (the SVMs were not evaluated, but could have been tested with each single subject classified against the other nine).

6.5 Training and Testing Groups

Careful selection of training and testing groups is an important step in the testing of classification problems. Three approaches were used for Hypotheses 1 and 2; these approaches are summarized in Table 6.6. A fourth technique, called cross validation, was used for Hypothesis 3.

TABLE 6.6 Training and testing groups

	Moniker	Number	Testing Sets	Training Sets
1	Leave out entire subjects	N_s	An entire subject	All other subjects
2	Leave out entire gait types	3	1) Free gait 2) Distracted gait 3) Paced gait	1) Distracted gait, paced gait 2) Free gait, paced gait 3) Free gait, distracted gait
3	Modified leave one out.	9	One of each gait type for each subject	All remaining samples

The first training and testing group involved using each subject as a testing set. The number of training sets and testing sets in this group was equal to the total number of subjects, N_s , in the classification problem; $N_s=10$ for all tests, except for the SVM two-class tests

for Hypothesis 1 (Hypothesis 1.2: $N_s=7$, Hypothesis 1.3: $N_s=5$, Hypothesis 1.4: $N_s=8$). This grouping was selected to investigate the robustness of the classifier, to see whether a classifier of, for example, "normal" gait could classify any subject without requiring a sample of that subject's gait during training.

The second training and testing group involved using a type of gait as the test set. As described in Appendix B.1, the gait trials for each subject were classified as "free gait," where the subject was told to walk as though she was taking a brisk walk through the park, "distracted gait," where the subject was given a task which was designed to provide distraction, and "paced gait," where a metronome was set at 120 clicks per minute, and the subject was asked to walk at a pace of one step per click. Because the purpose of the "distracted gait" was to draw out gait abnormalities in the subjects with PD, and the purpose of the "paced gait" was to improve the gait in the subjects with PD, these groupings were selected to see whether distracted PD gait was less likely to be misclassified and paced PD gait was more likely to be misclassified. The number of training sets and testing sets in this group was equal to three, for each of the three types of gait.

The third training and testing group was a modified version of "leave one out." Leave one out is generally considered one of the more accurate methods for analyzing classification results when only a relatively small amount of training data is available, because it maintains separation between the training and testing sets, while maximizing the (total) number of samples available to the training set [106]. In true "leave one out," the test set is a single sample from all of the data, and the training set is the rest of the data; the classification is trained as many times as the total number of samples. Because three of the techniques used were not readily automatable (only the SVM, a Matlab package, could easily be set to run through all the training and testing sets, providing all results after completion), it was not deemed feasible¹ to do this across all 193 samples for this initial evaluation of the

1. This should not be viewed as a long-term limitation; after a technique is identified as one which works well with this data, modifications to these implementations, or even new implementations, can certainly be developed.

various techniques. A modified version was developed, which involved filling the test set with three samples from each subject, with one sample of each gait type (free, distracted and paced). Each subject had between 4 and 9 trials of each gait type, so nine sets of testing and training data were set up. An algorithm¹ was developed to randomly place the trials into testing and training groups; for subjects which had fewer than nine trials of a particular gait type, the existing trials were randomly selected to fill all nine training and testing groups. The overall error is calculated from the sum of the errors for each of the nine groups. These groups were tested using all four techniques; the same groups were used for all techniques and all hypotheses.

Evaluation of Hypothesis 3 used a technique called cross validation for evaluation. For both CART, and the Weka implementation of Naïve Bayes and Neural Nets, a 10-fold cross evaluation was used (this simplified evaluation of this hypothesis, since these three techniques were not readily automatable). In a 10-fold cross validation, the samples are split into ten subsets of approximately equal size, with each of these ten subsets each used as a test set. The training is completed ten times and the overall error is completed from sum of the errors for the ten groups. This is quite similar to the modified leave one out used in Hypotheses 1 and 2, except, with the implementations used, there is no way to ensure that each of the ten subgroups has an equal representation of subjects, so it may be less accurate than the modified leave one out method.

6.6 Results

This section presents the results of each of the three hypotheses, using the various training and testing sets, and the four different techniques. The results are also detailed in Appendix C.2, including all the individual results for the nine groups used in the "modified leave one out" training and testing groups (this section presents the summed results).

1. Genleaveoutids.m, an m-file available in Appendix F.2, was used to generate the testing and training groups.

An additional classification with Neural Nets, and useful features indicated through use of CART are also presented.

6.6.1 Hypothesis 1

Hypothesis 1 was tested with three classes, using CART, Naïve Bayes, and a Neural Net, and with three sets of two classes, using SVMs. This hypothesis was designed to show that the two groups of females with normal gait were very similar. The hypothesis investigated the separation between the three classes, to see whether the NLF group were more similar to the NLFY group than the PDF group (e.g. the features were not classifying based on age-related parameters), and to support combining the NLF and NLFY in Hypothesis 2.

Three classes

The results for the "modified leave one out" training and testing sets, and tested on CART, Naïve Bayes, and a Neural Net, are shown in Tables 6.7, 6.8, and 6.9. The tables show the summed classification over the nine training/testing groups, the percentage of correct classifications, and the percentage classified as either NLFY or NLF. The results show that all three techniques are remarkably good at classifying the three classes of data. In the CART and Neural Net results, there is more confusion between the NLFY and the NLF classes than with the PDF class. The degree of confusion is not very high, but that is likely due to the fact that the "modified leave one out" training sets contain enough information about each subject to build the classifiers well. The Naïve Bayes results show close to even confusion between all three classes, but still with acceptable classification rates.

TABLE 6.7 CART Hypothesis 1 results, using modified leave one out

	NLFY	NLF	PDF	% correct	% NLFY or NLF
NLFY	122	10	3	90.4	97.8
NLF	6	47	1	87.0	98.2
PDF	0	3	78	96.3	3.7
Overall:				91.5	

TABLE 6.8 Naïve Bayes Hypothesis 1 results, using modified leave one out

	NLFY	NLF	PDF	% correct	% NLFY or NLF
NLFY	129	2	4	95.6	97.0
NLF	2	50	2	92.6	96.3
PDF	0	3	78	96.3	3.7
Overall:					95.2

TABLE 6.9 Neural Net Hypothesis 1 results, using modified leave one out

	NLFY	NLF	PDF	% correct	% NLFY or NLF
NLFY	131	2	2	97.0	98.5
NLF	3	50	1	92.6	98.2
PDF	0	0	81	100.0	0.0
Overall:					97.0

CART was also used to test this hypothesis using "leave one subject out" training and testing groups. The results, summed over the individual training and testing groups, are shown in Table 6.10, and shown by subject in Table 6.11. From Table 6.10, it is clear that there is significantly more confusion between the NLFY and NLF classes, than between either NLFY and PDF, or between NLF and PDF. While nearly a third of the NLFY samples were classified incorrectly, and nearly 90% of the NLF samples were classified incorrectly, only 5 out of 26 of the misclassified NLFY samples were misclassified as PDF, and only 1 out of 33 of the misclassified NLF samples were misclassified as PDF. These results demonstrate that when the classifier is tested with subject data not included in the training set, subjects with normal gait are more similar to each other, regardless of age.

Close to 30% of the PDF samples were misclassified (in Table 6.10), but a closer inspection of the results by subject, in Table 6.11, reveals that all of these misclassifications were samples from a single PDF subject. One of the limitations of this initial work is the small number of subjects available for evaluation. Subjects PDF-A and PDF-B appear to have similarities between their gait, because when either of these are used as the test sample,

the correct classification rate is 100.0%. However, the lower correct classification rate of PDF-C (or, rather, the misclassification of PDF-C as NLF and NLFY) suggests that this subject does not have similar feature values when compared with PDF-A or PDF-B. The effect of a small number of subjects in a group can also be seen in the results for the NLF, group, as only 4 samples of NLF-A and no samples of NLF-B were classified as NLF when each subject was the testing group.

TABLE 6.10 CART Hypothesis 1 results, by class, using leave one subject out

	NLFY	NLF	PDF	% correct	% NLFY or NLF
NLFY	66	21	5	71.7	94.6
NLF	32	4	1	10.8	97.3
PDF	4	13	47	73.4	26.7
Overall:					60.6

TABLE 6.11 CART Hypothesis 1 results, by subject, using leave one subject out

	NLFY	NLF	PDF	% correct	% NLFY or NLF
NLFY-A	16	0	0	100.0	100.0
NLFY-B	8	9	0	47.1	100.0
NLFY-C	10	9	2	47.6	90.5
NLFY-D	18	0	0	100.0	100.0
NLFY-E	14	3	3	70.0	85.0
NLF-A	15	4	0	21.1	100.0
NLF-B	17	0	1	0.0	94.4
PDF-A	0	0	20	100.0	0.0
PDF-B	0	0	23	100.0	0.0
PDF-C	4	13	4	19.1	81.0
Overall:					60.6

CART was also used with the testing groups consisting of a single type of gait (free gait, distracted gait, or paced gait), and the results for each of the three testing groups are

shown in Tables 6.12, 6.13, and 6.14. These testing groups were selected to investigate the effects of distraction and pacing on the gait of the PDF subjects. The distracted gait did have only a single misclassified sample, as compared to three with the free gait test set, suggesting that the distractions may have been successful at eliciting abnormalities in the gait of the PDF subjects. However, all of the PDF paced gait were correctly classified, suggesting that the pacing did not provide restorative feedback. An unusual result is the misclassification of seven samples of NLFY free gait as PDF; this may be a result of the technique used, as this was not seen in similar tests using the SVM, discussed below.

TABLE 6.12 CART Hypothesis 1 results, using "free gait" as the test set

	NLFY	NLF	PDF	% correct	% NLFY or NLF
NLFY	29	2	7	76.3	81.6
NLF	4	8	1	61.5	92.3
PDF	1	2	18	85.7	14.3
Overall:					76.4

TABLE 6.13 CART Hypothesis 1 results, using "distracted gait" as the test set

	NLFY	NLF	PDF	% correct	% NLFY or NLF
NLFY	27	0	3	90.0	90.0
NLF	1	14	0	93.3	100.0
PDF	1	0	24	96.0	4.0
Overall:					92.9

TABLE 6.14 CART Hypothesis 1 results, using "paced gait" as the test set

	NLFY	NLF	PDF	% correct	% NLFY or NLF
NLFY	19	2	3	79.2	87.5
NLF	0	9	0	100.0	100.0
PDF	0	0	18	100.0	0.0
Overall:					90.2

Two Classes

The results for each of the two class versions of Hypothesis 1, using "modified leave one out," and tested on SVMs, are shown in Tables 6.15, 6.16, and 6.17. The tables show the summed classification over the nine training/testing groups and the percentage of correct classifications.

The SVMs are reasonably good at classifying the different groups of data. As expected, there is some confusion between the NLFY and the NLF groups, as seen in Table 6.15. However, there is also confusion between the NLFY and the PDF groups, as seen in Table 6.17, and the most misclassification across all three versions is of NLF subjects as PDF, as seen in Table 6.16.

TABLE 6.15 SVM Hypothesis 1.1 results, using modified leave one out

	NLFY	NLF	% correct
NLFY	130	5	96.3
NLF	4	50	92.6
Overall:			95.2

TABLE 6.16 SVM Hypothesis 1.2 results, using modified leave one out

	NLF	PDF	% correct
NLF	46	8	85.2
PDF	1	80	98.8
Overall:			93.3

TABLE 6.17 SVM Hypothesis 1.3 results, using modified leave one out

	NLFY	PDF	% correct
NLFY	132	3	97.8
PDF	3	78	96.3
Overall:			97.2

To further evaluate these misclassifications, it is useful to look at the results from the "leave one subject out" tests, shown in Tables 6.18, 6.19, and 6.20. These results show that a sample in the PDF group was nearly equally likely to be misclassified as an NLF subject (50% misclassified), as it was to be misclassified as an NLFY subject (43.7%). As seen in the CART results earlier in this section, the classification rates of the NLF group, which had a total of two subjects were very low; 91.9% were misclassified as NLFY and 73% were misclassified as PDF. This second result is particularly interesting for the support of Hypothesis 1. Table 6.11 and Table 6.18 indicate that the gait of NLF-A and of NLF-B each have more similarities with the group of five NLFY subjects than they have similarities with each other. However, Table 6.19 indicates that when the only choice for classification of NLF-A or NLF-B is NLF, trained on only the opposite subject, or PDF, a higher percentage are classified correctly.

TABLE 6.18 SVM Hypothesis 1.1 results, using leave one subject out

	NLFY	NLF	% correct
NLFY	80	12	87.0
NLF	34	3	8.1
Overall:			64.3

TABLE 6.19 SVM Hypothesis 1.2 results, using leave one subject out

	NLF	PDF	% correct
NLF	10	27	27.0
PDF	32	32	50.0
Overall:			41.6

TABLE 6.20 SVM Hypothesis 1.3 results, using leave one subject out

	NLFY	PDF	% correct
NLFY	89	3	96.7
PDF	28	36	56.3
Overall:			80.1

SVMs were also used with testing groups consisting of one type of gait, and the full results are included in Appendix C.2. Of particular interest were Tables 6.21 and 6.22, which show the results of the PDF subjects compared with NLF and with NLFY, with the paced gait subset. Again, none of the PDF samples were misclassified, which suggests that the pacing may not have been effective at restoring the gait of the PDF subjects (there were PDF misclassifications with the distracted gait testing set and the free gait testing set).

Also of interest is Table 6.23. Above, the CART testing of Hypothesis 1 with the free gait testing set showed the unusual result of 7 NLFY samples misclassified as PDF. However, with SVM, only one of the NLFY samples was misclassified as PDF with the same testing set, suggesting that the CART result was due to the technique, rather than the actual data samples.

TABLE 6.21 SVM Hypothesis 1.2 results, using "paced gait" as the test set

	NLF	PDF	% correct
NLF	8	1	88.9
PDF	0	18	100.0
Overall:			96.3

TABLE 6.22 SVM Hypothesis 1.3 results, using "paced gait" as the test set

	NLFY	PDF	% correct
NLFY	21	3	87.5
PDF	0	18	100.0
Overall:			92.9

TABLE 6.23 SVM Hypothesis 1.3 results, using "free gait" as the test set

	NLFY	PDF	% correct
NLFY	37	1	97.4
PDF	1	20	95.2
Overall:			96.6

6.6.2 Hypothesis 2

The results for Hypothesis 2, using "modified leave one out," and tested on CART, SVMs, Naïve Bayes, and a Neural Net, are shown in Tables 6.24, 6.25, 6.26, and 6.27. The tables show the summed classification over the nine training/testing groups, and the percentage of correct classifications. The results for all classifications are very good, with correct classifications better than 95% for both categories using all four techniques. The standout technique is the Neural Net, which had 100% correct classifications for all PDF samples, and misclassified a mere 2 (1.1%) of NLF and NLFY samples.

TABLE 6.24 CART Hypothesis 2 results, using modified leave one out

	NLF/NLFY	PDF	% correct
NLF/NLFY	181	8	95.8
PDF	1	80	98.8
Overall:			96.7

TABLE 6.25 SVM Hypothesis 2 results, using modified leave one out

	NLF/NLFY	PDF	% correct
NLF/NLFY	181	8	95.8
PDF	2	79	97.5
Overall:			96.3

TABLE 6.26 Naïve Bayes Hypothesis 2 results, using modified leave one out

	NLF/NLFY	PDF	% correct
NLF/NLFY	181	8	95.8
PDF	4	77	95.1
Overall:			95.6

TABLE 6.27 Neural Net Hypothesis 2 results, using modified leave one out

	NLF/NLFY	PDF	% correct
NLF/NLFY	187	2	98.9
PDF	0	81	100.0
Overall:			99.3

The results for Hypothesis 2, using "leave one subject out," and tested on CART, SVMs, and a Neural Net are shown in Tables 6.28, 6.29, and 6.30. These results are cautionary: though the combined NLF/NLFY group still has reasonable rates of classification, particularly with CART, the PDF group has significantly lower rates of correct classification with both techniques. As seen previously, this is likely due to the small number of subjects (only three PDF), so an important next step will be to collect data from more subjects.

TABLE 6.28 CART Hypothesis 2 results, using leave one subject out

	NLF/NLFY	PDF	% correct
NLF/NLFY	123	6	95.3
PDF	21	43	67.2
Overall:			86.0

TABLE 6.29 SVM Hypothesis 2 results, using leave one subject out

	NLF/NLFY	PDF	% correct
NLF/NLFY	114	15	88.4
PDF	33	31	48.4
Overall:			75.1

TABLE 6.30 Neural Net Hypothesis 2 results, using leave one subject out

	NLF/NLFY	PDF	% correct
NLF/NLFY	122	7	94.6
PDF	18	46	71.9
Overall:			87.0

Hypothesis 2 was also tested using the testing groups consisting of one type of gait, on CART and SVMs. The results, included in Appendix C.2, are similar to the results seen when these testing groups were used with Hypothesis 1.

6.6.3 Hypothesis 3

Hypothesis 3 was tested using ten-fold cross-validation, on CART, Naïve Bayes, and a Neural Net. The CART and Naïve Bayes results are in Appendix C.2, and the Neural Net results are shown in Table 6.31, including the summed classification over the ten subsets of training/testing groups, the percentage of correct classifications, and the percentage classified as either NLFY or NLF.

TABLE 6.31 Neural Network Hypothesis 3 results, by subject, using cross-validation

	NLFY A	NLFY B	NLFY C	NLFY D	NLFY E	NLF A	NLF B	PDF A	PDF B	PDF C	% correct	% NLFY or NLF
NLFY-A	15	0	0	1	0	0	0	0	0	0	93.8	100.0
NLFY-B	0	17	0	0	0	0	0	0	0	0	100.0	100.0
NLFY-C	0	0	20	0	1	0	0	0	0	0	95.2	100.0
NLFY-D	0	0	0	18	0	0	0	0	0	0	100.0	100.0
NLFY-E	0	0	0	0	18	1	0	0	0	1	90.0	95.0
NLF-A	0	0	0	0	0	19	0	0	0	0	100.0	100.0
NLF-B	0	0	0	0	0	0	18	0	0	0	100.0	100.0
PDF-A	0	0	0	0	0	0	0	20	0	0	100.0	0.0
PDF-B	0	0	0	0	0	0	0	0	23	0	100.0	0.0
PDF-C	0	0	0	0	0	0	0	1	0	20	95.2	0.0
Overall:										97.4		

Though all three techniques performed well, the Neural Network again had the best results. Only five samples out of 193 were misclassified with the Neural Network, and only one of those was classified as a subject of a different class. This high classification rate suggests that the GaitShoe may be highly capable of capturing the nuances of individual subjects' gait (see Section 6.6.5 for Neural Net results using fewer features).

6.6.4 CART Feature Information

As discussed earlier in this chapter, one of the most powerful aspects of CART is the transparency of the trees as to which features are the most informative. All of the decision trees resulting from the tests done on the three hypotheses were investigated, and the features which were most frequently used to split nodes are listed in Table 6.32, in decreasing order of frequency (the feature number corresponds to the number in Table 6.4); these reduced number of features will be used to train a Neural Net in the following section.

TABLE 6.32 Informative features, as identified by CART

Feature Number	Feature Derivation	Feature Description
1	Standard deviation of FSRsum/bodyweight, L and R combined	Walking energy variation
21	Mean minimum pitch, L and R combined	Shuffle index
3	Mean StepF/bodyweight, L and R combined	Step energy amplitude
24	Mean percent stance time, L and R combined	Shuffle duration
10	Gyro-z variation, L and R combined	Pitch variation
20	Maximum pitch variation, L and R combined	Shuffle variation
8, 6	Gyro-y, -x variation, L and R combined	Roll, yaw variation
12, 14, 16	Accel-x, -y, -z variation, L and R combined	Linear motion variation

Of particular interest is the fact that all of these features used the metric "L and R combined," which simply combined the data from the left and right feet into a single vector before the analysis was applied. This result suggests that only a single shoe may be necessary to capture the differences between the gait of subjects with PD and subjects with normal gait. Of course, certain pathologies, such as cerebral vascular accidents (strokes), known to result in asymmetrical gait would likely still benefit from data from both shoes, so that the metrics with the ratio between the left foot data and the right foot data could be calculated. However, for non-asymmetric pathologies, using only half of the GaitShoe system would greatly simplify the data collection process.

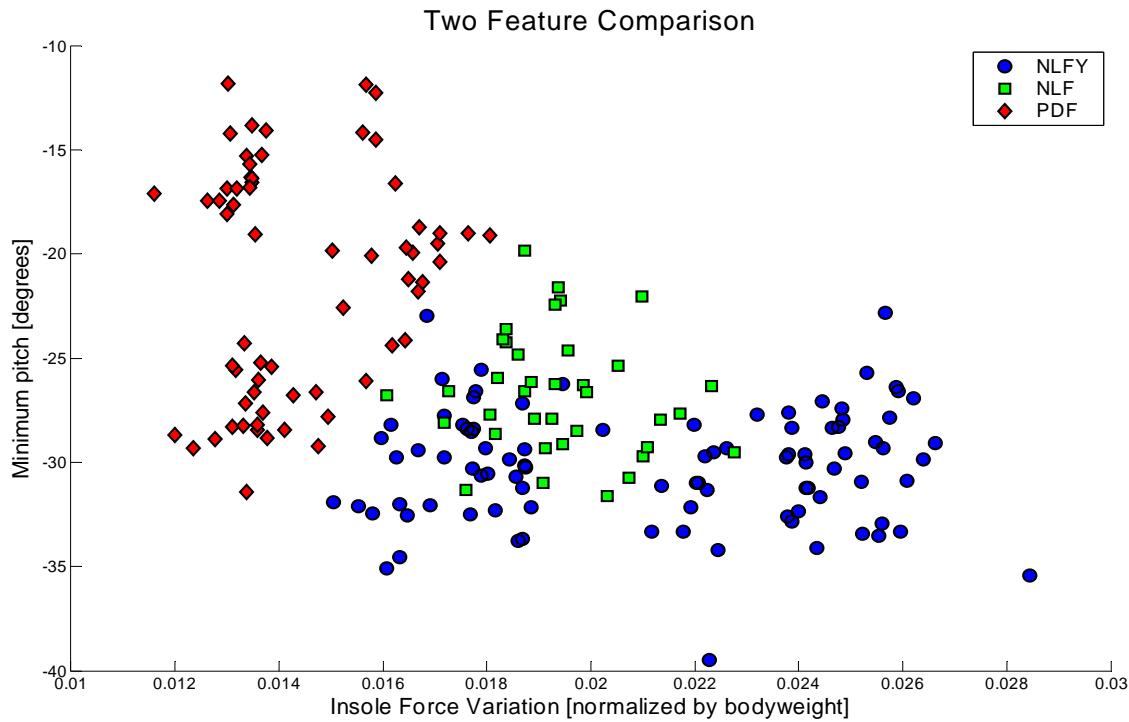


Figure 6.3 Subject data of two most informative features in CART analyses

The first two features listed in Table 6.32 occurred far more frequently than the other features. These two features, the mean minimum pitch, and the variation in the sum of the four FSRs, are plotted for all of the data samples in Figure 6.3. This graph shows an impressive separation of the PDF class from the NLFY and NLF classes. In addition, it shows considerable confusion between the NLFY and NLF classes, suggesting that these two features are not simply separating the PDF samples on the basis of an age-related parameter.

Feature 1 is plotted along the x-axis, and was derived by taking the clipped standard deviation of the sum of all four FSRs, for both the left and right feet data. This feature is a measure of the variance of the force measured underneath the foot (and normalized by body weight). The magnitude of the total force seen between the foot and the floor is typically 120% of body weight in normal walking gait, but reaches as much as 220% of body weight during running [105]. Therefore, a larger variance (when normalized by body-

weight) corresponds to a more "energetic" gait, so this feature can be referred to as a measure of "walking energy variation." Figure 6.3 demonstrates that the subjects with PD have a much less energetic gait.

Feature 21 is plotted along the y-axis, and was derived by calculating the mean of the minimum pitch seen during gait. As shown in Figure 4.16 (p. 120), the minimum pitch occurs just before heel strike, and corresponds to both the amount of leg swing, the gait velocity, and the amount of dorsiflexion of the foot. As both leg swing and dorsiflexion are likely to be greatly reduced when the feet are shuffled, this feature can be described as a "shuffle index." Figure 6.3 shows that subjects with PD are more likely to have a lower magnitude minimum foot pitch, or a higher "shuffle index".

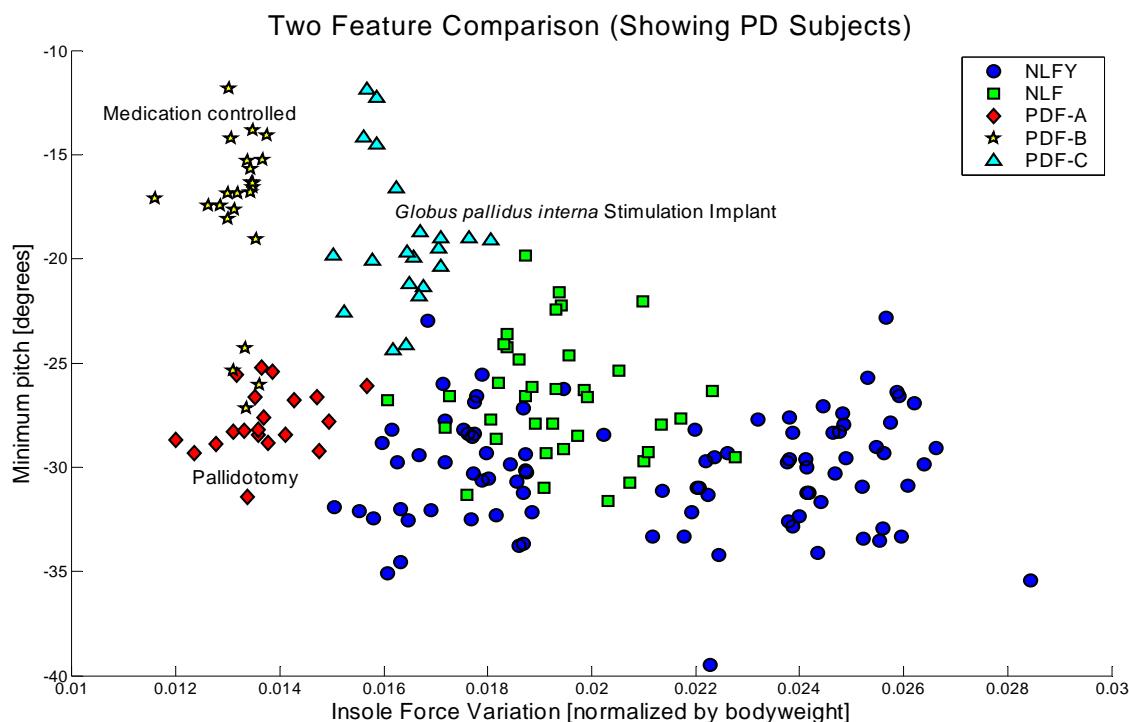


Figure 6.4 Subject data of two most informative features in CART analyses, individual PD subjects

Figure 6.4 is the same graph as Figure 6.3, but with the individual PD subjects identified, and labeled by clinical treatment. Interestingly, the two PD subjects that had surgical interventions are closer to the NLFY and NLF groups, with the PD subject treated only with medication is further away. With only three subjects, these results cannot be extrapolated

to the PD population at large, however, it will be of great interest to see whether this result holds as more PD subjects, with various interventions, are tested and included in the classification.

6.6.5 Additional Neural Net Studies

The features identified by the CART, as described in Section 6.6.4, were used to train Neural Nets using reduced numbers of features, with 10-fold cross-validation.

For Hypothesis 1, when only the top two features, the mean minimum pitch ("shuffle index") and the insole force variation ("walking energy variation") were used, the overall classification was 86.5%, as shown in Table 6.33. However, this included a large number of misclassification between NLFY and NLF, which is not surprising given Figure 6.3, and only three misclassifications between NLFY/NLF and PDF.

TABLE 6.33 Neural Net Hypothesis 1 results, using 10-fold cross-validation, and the top two features

	NLFY	NLF	PDF	% correct	% NLFY or NLF
NLFY	85	6	1	92.4	98.9
NLF	17	19	1	51.4	97.3
PDF	1	0	63	98.4	1.6
Overall:				86.5	

For Hypothesis 2, the results, shown in Table 6.34, were excellent when the top two features were used. The overall classification rate was 99%, with just a single sample from each class misclassified.

TABLE 6.34 Neural Net Hypothesis 2 results, using 10-fold cross-validation, and the top two features

	NLF/NLFY	PDF	% correct
NLF/NLFY	128	1	99.2
PDF	1	63	98.4
Overall:		99.0	

The results for Hypothesis 3 are shown in Table 6.35, using the top six features, the mean minimum pitch ("shuffle index"), the insole force variation ("walking energy variation"), mean force per step ("step energy amplitude"), mean percent stance time ("shuffle duration"), z-gyroscope variation ("pitch variation"), and maximum pitch variation ("shuffle variation").

TABLE 6.35 Neural Net Hypothesis 3 results, by subject, using 10-fold cross-validation, and the top six features

	NLFY A	NLFY B	NLFY C	NLFY D	NLFY E	NLF A	NLF B	PDF A	PDF B	PDF C	% correct	% NLFY or NLF
NLFY-A	15	1	0	0	0	0	0	0	0	0	93.8	100.0
NLFY-B	1	14	0	2	0	0	0	0	0	0	82.4	100.0
NLFY-C	1	0	19	0	0	0	0	0	1	0	90.5	95.2
NLFY-D	0	0	0	18	0	0	0	0	0	0	100.0	100.0
NLFY-E	0	0	1	0	17	2	0	0	0	0	85.0	100.0
NLF-A	0	0	0	0	3	16	0	0	0	0	84.2	100.0
NLF-B	0	0	0	1	0	0	17	0	0	0	94.4	100.0
PDF-A	0	0	1	0	0	0	0	18	1	0	90.0	5.0
PDF-B	0	0	0	0	0	0	0	1	22	0	95.7	0.0
PDF-C	0	0	0	0	0	0	0	0	0	21	100.0	0.0
Overall:											91.7	

The overall classification rate for Hypothesis 3 is 91.7%, with only two misclassification between NLFY/NLF and PDF. This is an interesting result, because there are fewer features than subjects (6 features, 10 subjects), as compared to the results in Section 6.6.3, where there were more features than subjects; one could argue that with sufficient number of features, any subject could be identified. However, this result demonstrates that, for this group of subjects, these six features were enough to classify these ten subjects with a classification rate better than 90%.

Finally, a Neural Net was trained on three different sets of "contrived" groups. With the small number of subjects, and with the strong results for Hypothesis 3, the question arises whether the strong classification rate between NLF/NLFY and PDF is simply the result of the Neural Net training on subjects, rather than on group similarity. Although Figure 6.3 indicates that there is actual separation between the NLF/NLFY and PDF groups, three "contrived" groupings were set up, each with one subgroup of seven subjects, and a second subgroup of three subjects. The groupings, shown in Table 6.36, were selected such that each subgroup of three subjects had one of the three PDF subjects, and such that the two NLF subjects were each in a subgroup of three, as well. In addition, because just three contrived groupings were set up for this quick investigation (rather than an exhaustive analysis of all combination of seven and three subgroups), the groups of three subjects were set up by inspection of the data tables from all the previous classification, such that subjects who had been misclassified as each other were included in the same subgroup.

TABLE 6.36 Contrived Groupings

	7 Subject Subgroup	3 Subject Subgroup
Group A	NLFY-A, NLFY-B, NLFY-C, NLFY-D, NLF-A, PDF-A, PDF-B	NLFY-E, NLF-B, PDF-C
Group B	NLFY-A, NLFY-B, NLFY-D, NLFY-E, NLF-B, PDF-A, PDF-C	NLFY-C, NLF-A, PDF-B
Group C	NLFY-B, NLFY-C, NLFY-E, NLF-A, NLF-B, PDF-B, PDF-C	NLFY-A, NLFY-D, PDF-A

The results are shown in Tables 6.37, 6.38, and 6.39, and have classification rates of 79.3%, 68.4%, 80.8%. Compared with the 99% classification result shown in Table 6.34, for groups NLFY/NLF and PDF, this suggests that the Neural Net is indeed able to train on group characteristics rather than on individual characteristics...

TABLE 6.37 Neural Net results, using 10-fold cross-validation,
and the top two features, with "Contrived Groups A"

	Group A-1	Group A-2	% correct
Group A-1 (7 subjects)	118	16	88.1
Group A-2 (3 subjects)	24	35	59.3
Overall:			79.3

TABLE 6.38 Neural Net results, using 10-fold cross-validation,
and the top two features, with "Contrived Groups B"

	Group B-1	Group B-2	% correct
Group B-1 (7 subjects)	108	22	83.1
Group B-2 (3 subjects)	39	27	38.1
Overall:			68.4

TABLE 6.39 Neural Net results, using 10-fold cross-validation,
and the top two features, with "Contrived Groups C"

	Group C-1	Group C-2	% correct
Group C-1 (7 subjects)	124	15	89.2
Group C-2 (3 subjects)	22	32	59.3
Overall:			80.8

6.7 Discussion

This chapter investigated the ability of four classic pattern recognition techniques to distinguish gait using features derived from the vast quantity of information measured by the GaitShoe.

The primary goal was to classify the gait of subjects with Parkinson's disease from the gait of subjects with normal gait, to see whether the GaitShoe sensor measurements encapsulated information about changes in gait between the two groups. This type of information could be used to evaluate treatment strategies for patients with Parkinson's disease.

The subjects were selected from the fifteen volunteers who were subjects for the validation of the GaitShoe described in Chapter 5. The goal of the subject selection was to create classes that differed only on the basis of the presence of Parkinson's disease. Only females were used, both to eliminate any question of gender differences, and because the females had a better match of ages. Though the ages ranged from early twenties to mid-sixties, the group of ten female subjects had a subset of two subjects with normal gait who were reasonably age-matched to the three subjects with Parkinson's disease. In addition, the

females had a small variation in height, suggesting that leg length was unlikely to affect the outcome of the classification.

The first hypothesis examined three classes: NLFY, females with normal gait who were under 30 years old; NLF, females with normal gait who were 48 and 54 years old; and, PDF, females with Parkinson's disease who were over 54 years old. The goal of this hypothesis was to see if the features selected showed significantly more discriminating among PDF and NLF classes than among NLFY and NLF classes. This was well demonstrated by the CART and SVM models built with the training and testing sets where individual subjects were used as the testing set. Samples from the NLFY and NLF classes were far more likely to be misclassified as NLF or NLFY, respectively, than as PDF.

The second hypothesis used the positive result from the first hypothesis to combine all the NLFY and NLF subjects into one group, NLFY/NLF. The four techniques were then used to classify samples as either NLFY/NLF or PDF, and all techniques performed very well, with correct classifications better than 95%. The strongest result was from the Neural Net, which correctly classified all of the PDF samples, and only misclassified 2 (1.1%) of the combined NLF and NLFY samples. In addition, when trained using only the top two features, the mean minimum pitch ("shuffle index") and the insole force variation ("walking energy variation"), the Neural Net classification rate was 99%, with only two samples misclassified. These results suggest that the GaitShoe is quite capable of capturing changes in the gait due to Parkinson's disease.

The third hypothesis investigated whether subjects could be classified individually, using CART, Naïve Bayes and Neural Nets, all of which could handle multiple classes, to classify the ten different subjects. Again the Neural Net results were outstanding, with only five out of 193 samples misclassified. The Neural Net was trained on only six features, the mean minimum pitch ("shuffle index"), the insole force variation ("walking energy variation"), mean force per step ("step energy amplitude"), mean percent stance time ("shuffle duration"), z-gyroscope variation ("pitch variation"), and maximum pitch variation ("shuf-

file variation"), and was able to classify the ten subjects with a classification rate of 91.7%. These results suggest that the features derived from the GaitShoe measurements are able to capture the individualities of each subject's gait.

The Naïve Bayes classification results tended to be the weakest. However, Naïve Bayes requires the assumption that the features were independent, which was certainly not true for the features used here, and likely had an impact on the classification. The SVM classification results were reasonable, but as use of SVMs are limited to two classes, the applicability to larger groups of gait subjects is limited, though multiple classes can be evaluated by developing SVMs with one of the classes evaluated against the others (further work with SVMs, including adjusting the settings may result in better SVM classification rates).

Though the CART classification results were reasonable, the real benefit of the CART analysis was in the identification of two features which provide excellent separation between the subjects with Parkinson's disease and the subjects with normal gait. As discussed above, the top CART-identified features were used to run additional Neural Net analyses to demonstrate that these subjects can be classified with a small number of features.

The Neural Net classification results were consistently the strongest, and demonstrate great promise for future use in using the GaitShoe system to classify both individual gaits as well as the gaits of groups of subjects.

The overall results from the pattern recognition analysis suggest that use of methods such as a Neural Net in combination with the features of the GaitShoe may have great benefit in analyzing gait.

Chapter 7

REAL-TIME THERAPEUTIC FEEDBACK

One application for the GaitShoe would use real-time analysis of the sensor outputs to provide feedback about the current gait, to allow the user to make adjustments. This could be useful for many areas, such as physical therapy, sports medicine, or athletic training, and the feedback could take a variety of forms: musical, tonal, visual, tactile. It could also be used to provide electro-stimulation at certain parts of the gait cycle [13], or to control an artificial leg. This chapter describes an initial investigation into using musical feedback controlled by a real-time analysis of the GaitShoe sensor data [107].

7.1 Overview

Music Therapy is an established field; however, it generally consists of patients listening to a specific type of music, or patients playing musical instruments [108]. Even so, the idea of using physiological measurements to control electronic music has been explored for some time, notably the work by David Rosenboom and Richard Teitelbaum in the late 1960s, involving the use of brainwaves, heartrates, EMGs, and skin conductivity to produce real-time musical biofeedback [109]. Other recent work involves interactive music and visuals set up as a meditation chamber, which responded to measurements of galvanic skin response, respiratory rate, and heart rate [110], or interactive music with causal mappings between free gesture and sound to encourage withdrawn mentally disabled and autistic children to become engaged [111].

The use of on-body sensors for sports applications is becoming more common, but audio feedback is generally limited to a simple beep, or a critique of a golf or batting swing from a talking virtual coach [112]. However, applications in dance, such as the Expressive Footware, as described in Section 2.1.5, have made use of motion-to-music mappings to allow the dancer to control the music heard during the performance [38].

Lack of applications in both the physical therapy and sports fields is likely due at least in part to the absence of readily available methods for gathering and analyzing relevant physiological data in real-time. Thus, the heavily instrumented GaitShoe could open the door to many new applications of musical feedback.

7.2 Rhythmic Auditory Stimulator

To explore the use of the GaitShoe for real-time feedback, the "Rhythmic Auditory Stimulator" (RAS) program [113] was developed in conjunction with Erik Asmussen, an undergraduate researcher collaborating with our group. Using insight gained during the gait analysis described in Chapter 5, the RAS provided three different types of sensing and feedback. Mr. Asmussen is pictured next to the RAS running in Figure 7.1.



Figure 7.1 Erik Asmussen and the Rhythmic Auditory Stimulator

7.2.1 The RAS system

Mr. Asmussen wrote the interactive RAS environment using the Max/MSP graphical programming language [114] on an Apple Computers powerbook. The GaitShoe basestation was connected to the powerbook via a Keyspan 19HS USB Serial Adapter [115]. Software requirements within Max/MSP limited the serial data to rate to 56.6 kbps rather than the usual 115.2 kbps, which reduced the data transfer rate of each shoe to approximately 30 Hz. The musical feedback generated with Max/MSP was output from the powerbook using Musical Instrument Digital Interface (MIDI). MIDI is a standard that provides a method of easy transmission of information corresponding to electronic music. A Midiman USB MidiSport [116] was connected to the USB port of the powerbook, and transmitted the MIDI output to an E-MU Proteus 2000 synthesizer [117], which output the sound to speakers. This allowed the system to respond in real-time, approximately 100 ms after the gait event of interest.

Although the RAS program can be configured to use any sensor outputs produced by the GaitShoe, the tests discussed here only involved using the FSRsum parameter derived from the sum of the four FSRs, "FSRsum," which is described in detail in Section 4.9, and the paired sums of the two medial FSRs and of the two lateral FSRs. Screenshots for three of the menus in the RAS system are shown in Figure 7.2.

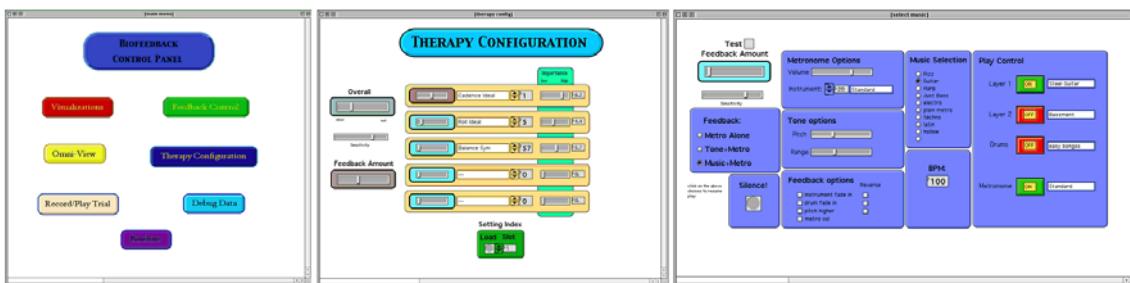


Figure 7.2 Screenshots of the RAS system, showing the main menu (upper left), therapy configuration menu (upper right), and feedback control menu (lower).

7.2.2 RAS Feedback

Three types of sensing were developed for the RAS: pace sensing, force distribution sensing, and peak force sensing.

Pace Sensing

This application was designed to aid subjects with Parkinson's disease. As discussed in Section 2.3.1 (p. 44), previous work has shown that rhythmic cues at a pace slightly faster than the PD subject's normal pace helps to lengthen stride and increase mean gait velocity. Previous work, however, has just involved a passive metronome [55] [56], so the goal of this application was to actively sense the pace of the subject, and provide feedback only as necessary.

The FSRsum was used to determine heel strike time, and the current pace was determined by subtracting the previous heel strike time from the current heel strike time. The RAS contained a field to set the ideal pace, and the current pace approached the ideal pace to determine the feedback.

Two different modes of feedback were available; the first provided the user with very subtle rhythmic cues, which faded out when current pace was equal to the ideal pace. The second mode played a charming tune while the current pace was at the ideal pace, with a quiet drumbeat at the ideal pace in the background, and when the current pace diverged from the ideal, the feedback converted to a loud drum sound only. When the user returned to the ideal pace, the charming tune returned as a reward. The first mode explored simple cues only when needed, while the second mode always produced background music (inspired by the ubiquity of portable music players in today's society), with changes in correspondence with the feedback.

Force Distribution Sensing

This application was an initial investigation into changes in force distribution. A gait parameter of interest for runners is the degree of pronation and supination, as excessive

pronation or supination can lead to injury if left uncorrected. This condition is typically treated by the placement of orthotics in the shoe, but feedback could allow the runner to try to correct the condition (for instance, if it occurs only as the runner becomes fatigued). Pronation is a complex motion involving changes in the motion of the ankle. However, excessive pronation or supination each have manifestations in the distribution of force underneath the foot; inspection of the underside of an older pair of running shoes usually reveals uneven wear patterns on the medial or lateral edge of the shoes, corresponding to over-pronation and supination, respectively.

The paired sums of the medial FSRs and the lateral FSRs during static standing were stored. During gait, the sum of the medial FSRs and the sum of the lateral FSRs were compared to the stored results, to see if an excessive amount of force was applied either medially or laterally. This mode of feedback played the user a charming tune in a major key when force distribution was determined to be within normal bounds. When excessive force was detected either medially or laterally, the tune transitioned from a major to a minor key, and if left uncorrected, became progressively dissonant.

Peak Force Sensing

The final application looked at the peak force detected by the FSRsum during stance. This application was designed to aid patients, for example, recovering from a broken leg, or after hip or knee replacement. During recovery, patients are told to apply a certain percentage of body-weight each week. The percentage gradually increases until the patient can walk normally again; generally, the patient is instructed to step on a scale to see what that week's weight limit feels like. Use of the GaitShoe system could provide the patient with real-time feedback as to the force applied on the recovering leg, and could remove an element of guess-work from the patient's recovery.

The RAS contained a field to set the acceptable threshold for force applied across the four FSRs. This mode of feedback was similar to the previous mode: when the force was under the threshold, the user heard the charming tune in a major key. If the threshold was

exceeded by a small amount, the tune transitioned to a minor key, and if exceeded by a large amount, the music became very dissonant.

7.3 Conclusions

This initial work demonstrated that the GaitShoe could be analyzed in real-time, and configured to provide real-time feedback to the user about a variety of changes in gait. There are many applications in both physical therapy and sports medicine which might benefit from this type of feedback. Videos of the three sensing modes described here are archived on-line, at <http://www.media.mit.edu/resenv/GaitShoe/index.html>. The next step for this work is to evaluate the feedback on patients; tests using the pace sensing for subjects with Parkinson's disease are planned.

Chapter 8

CONCLUSION

The development of the GaitShoe has resulted in a wireless wearable system with an unprecedented number of sensors designed to capture information that can characterize gait of both feet. The system costs under \$500 per foot in prototype quantities and the hardware for a single shoe weighs under 300 g. The hardware is readily fixed to a variety of typical walking shoes, and data can be continuously collected over a few hours.

The gait parameter analysis indicated that the GaitShoe can be further developed into a true wearable podiatric laboratory, which could be of great use in evaluating gait over longer periods of time than are available in motion laboratories, as well as allowing the evaluation to be carried out in a neutral environment, such as the subject's home. It would also allow the evaluation of subjects who are without access to a motion laboratory.

Relevant GaitShoe sensors were calibrated and analyzed to determine parameters of gait, which were validated by comparison with data collected simultaneously by the Massachusetts General Hospital (MGH) Biomotion Laboratory. The GaitShoe's determination of heel strike time regularly anticipated the time determined from analysis of the force plate output; the results suggest that the GaitShoe is capable of detecting heel strike before the force plate. The toe off times determined by the GaitShoe and from the force plate data were very similar. Placement of force sensitive resistors (FSRs) underneath the toe may result in even the ability to detect toe off timing more accurately than the force plate. This should be investigated further, by further calibration of the force sensors in order to deter-

mine appropriate thresholds corresponding to the initiation of loading of the FSRs; this needs to be done such that the time-scale of the GaitShoe can be precisely aligned with the time-scale of the calibration system. In addition, the four coarsely spaced FSRs provide a reasonable approximation to the force distribution measured by the force plate, and are capable of providing information about shifting weight patterns from stride to stride (information which is not available from a force plate).

A simplified analysis of the motion of the foot, using the x-accelerometer and the z-gyroscope, resulted in reasonable estimations of the pitch, velocity and stride length. However, a more complete analysis including the outputs of the x- and y-gyroscopes, and the outputs of the y- and x-accelerometers is expected to improve the results. In addition, the implementation of a Kalman filter is likely to further improve the outcome. Though future work should certainly make use of the full suite of gyroscopes and accelerometers, the results of the simplified analysis indicate that the GaitShoe is capable of reasonable estimations of orientation and displacement.

The bend sensor output generally has a shape corresponding to the expected plantar flexion and dorsiflexion curve. The output is likely to be more uniform with an improved method of positioning and/or retaining the sensor. Further evaluation should utilize an alternative reference system that allows the bend sensor to be easily held next to the ankle, and that has fewer errors. Alternatively, the use of a more repeatable bend sensor could be considered, such as one made from fiber optics.

The electric field sensor provides a method of determining the height of the foot above the floor. In particular, multiple discrete electric field sensors can be implemented, such as at the heel and the toe or metatarsals, which would provide additional information about the orientation of the foot. The recent development of an ultrasound sensor by Steven Dan Lovell provides a method of measuring the distance between the two feet and the relative orientation of the feet, as well as a (future) method to measure the height of the foot above the ground.

In addition, the overall insole design should be reconsidered for future work. The PVDF sensors were not used in the analysis, because of the variable output. The FSRs were very valuable, but a more precise pressure sensor or alternative implementations of the FSR might provide even better results. The electric field sensor connector must be replaced with one that can better accommodate the coaxial cable used to provide the direct signal shield around the connection to the electrode. The ultrasound sensor transmitter and receiver attachments need to be redesigned so that they are much more stable, and are not affected by dynamics of gait, such as the impact of heel strike.

The pattern recognition results suggested that a great future application for the GaitShoe may be the use of the GaitShoe's ability to derive meaningful features from the extensive sensor suite, and to use those features to recognize individual subjects as well as groups of subjects with a similar gait. In particular, Neural Nets appeared to be a very promising method for discriminating between both individual subjects and between groups of subject with normal gait, and groups of subjects with Parkinson's disease. The results should be confirmed with a broader study including larger numbers of subjects, as well as subjects of both genders. If the Parkinson's disease results remain strong once subjects are added, this technique may be able to be used to assess the effectiveness of the patient's medication regimen, or even to assess the impact of various treatments.

In addition, the use of Classification and Regression Trees (CART) provided insight into the most useful features for discriminating between the two groups. The standard deviation of the FSRsum, normalized by body weight ("walking energy variation") and the mean minimum pitch ("shuffle index") provided a excellent separation between the subjects with Parkinson's disease and the subjects with normal gait,. These results show that the subjects with Parkinson's disease are closer to a shuffle-gait than the normal subjects, and that their steps have less force than the steps of subjects with normal gait.

The GaitShoe system was incorporated with a program written by Erik Asmussen in the Max/MSP graphical programming language, and output MIDI (the standard for transmit-

ting electronic music) to a synthesizer to provide rhythmic auditory stimulation (RAS). The RAS implemented was real-time musical feedback corresponding to the detection of three gait conditions: stride pace, weight distribution, and total weight. The system worked well and provided interesting and engaging feedback; in particular, while filming videos of the GaitShoe and the RAS, people walking by could be observed moving their head and upper body in response to the catchy percussion rhythm used in the stride pace feedback. Future work should include testing the feedback on patients with relevant gait pathologies to evaluate the effectiveness of the feedback, and whether the feedback is interesting and engaging to those who would use it for physical therapy. This is an exciting area of future application for the GaitShoe, as it provides a new framework in which interactive real-time physical therapy can be investigated. This has many applications, from rehabilitating gait in patients such as those recovering from hip surgery who need to bear only a certain amount of weight on one leg, or those recovering from a stroke who need to relearn how to walk symmetrically, to investigating the gait of subjects with diabetic neuropathy to evaluate risk for ulceration, to countless applications in the sports medicine and sports training fields, such as detections of over-pronation or supination to provide runners with feedback to allow them to make changes in their running gait or to make decisions about when to stop a run if at risk for injury, to analysis.

Table 8.1 details a comparison between the GaitShoe and the MGH Biomotion Laboratory systems. While the GaitShoe is not yet a tool to use for evaluation of gait prior to surgical intervention, with the changes suggested in this chapter, it may be able to replace traditional motion laboratories for such clinical work as evaluation during design of orthotics and prosthetics. In particular, it will be an excellent tool for areas without access to motion laboratories. In addition, the GaitShoe has already been implemented in a simple real-time analysis and feedback system; a valuable possible application for the GaitShoe would be to use real-time analysis to provide electro-stimulation (for persons with spinal cord injury) or control of an artificial leg, at specific times during the gait cycle.

TABLE 8.1 Comparison between the GaitShoe and the MGH Biomotion Lab

GaitShoe	MGH Biomotion Lab
More than 30 degrees of freedom across lower legs and feet.	24 degrees of freedom on lower legs and feet.
No data collection on rest of body.	42 degrees of freedom on rest of body.
Collects data continuously.	Collects 7 seconds of data.
Collects data “anywhere”.	Collects data in BML lab only.
Wireless.	Tethered.
Can be analyzed in real-time.	Data processed after collection is completed.
Can be used to control real-time feedback.	Cannot be used for real-time control.
Total mass < 0.6 kg.	Total mass < 2 kg.
Cost < \$1K in prototype quantities.	Cost: Proprietary equipment \$1M, plus space and personnel costs for use of system (furnishing a lab with commercial equipment would be >\$250K)
Heel strike time determined with a $\sigma < 0.023$ sec and toe off time determined with a $\sigma < 0.017$ sec, as compared with BML.	Heel strike and toe off times determined within 0.007 sec.
Stride length determined with a $\sigma < 16$ cm, as compared with BML.	Distances, such as stride length, determined within 1 mm
Pitch determined with a $\sigma < 7^\circ$, as compared with BML.	Orientations, such as pitch, determined within 1°
Capable of classifying gait of groups and gait of individuals, on small subject sample.	

The GaitShoe is a research tool that enables the analysis of gait in untraditional ways, such as over long periods of time and in the home environment or through use of pattern recognition, and provides a method for real-time feedback for use in such applications as sports medicine, electro-stimulation, or physical therapy.

Appendix A

MEDICAL INFORMATION

A.1 Terminology

Calcaneous: The largest foot bone, located in the heel of the foot.

Dorsiflexion: Flexion of the foot; forefoot motion upward, toward the ankle.

Gait: Manner of walking.

Heel strike: Time at which the heel first makes contact with the floor; indicates end of swing and start of *stance*; see Figure A.1.

Lateral: Away from the centerline of the body; for the right foot, the right edge is lateral.

Medial: Toward the centerline of the body; for the right foot, the left edge is medial.

Metatarsal heads: The distal ends of the metatarsal bones, at the point of articulation with the proximal phalanx of the corresponding toe (located at the "ball of the foot").

Plantar flexion: Extension of the foot; forefoot motion downward, away from the ankle.

Pronation: Complex motion of the ankle, resulting in the sole of the foot shifting medially; occurs during the start of stance to absorb shock from heel strike and to assist in balance.

Stance: Period in which the foot is in contact with the floor; this generally takes up about 60% of the stride cycle; see Figure A.1.

Step: Interval between two successive heel strikes for opposite feet; see Figure A.1.

Stride: Interval between two successive heel strikes of the same foot; see Figure A.1.

Supination: Complex motion of the ankle, resulting in the sole of the foot shifting laterally.

Swing: Period during which the foot is not in contact with the floor; this generally takes up about 40% of the stride cycle; see Figure A.1.

Toe off: Time at which the great toe is first no longer in contact with the floor; indicates end of *stance* and start of *swing*; see Figure A.1.

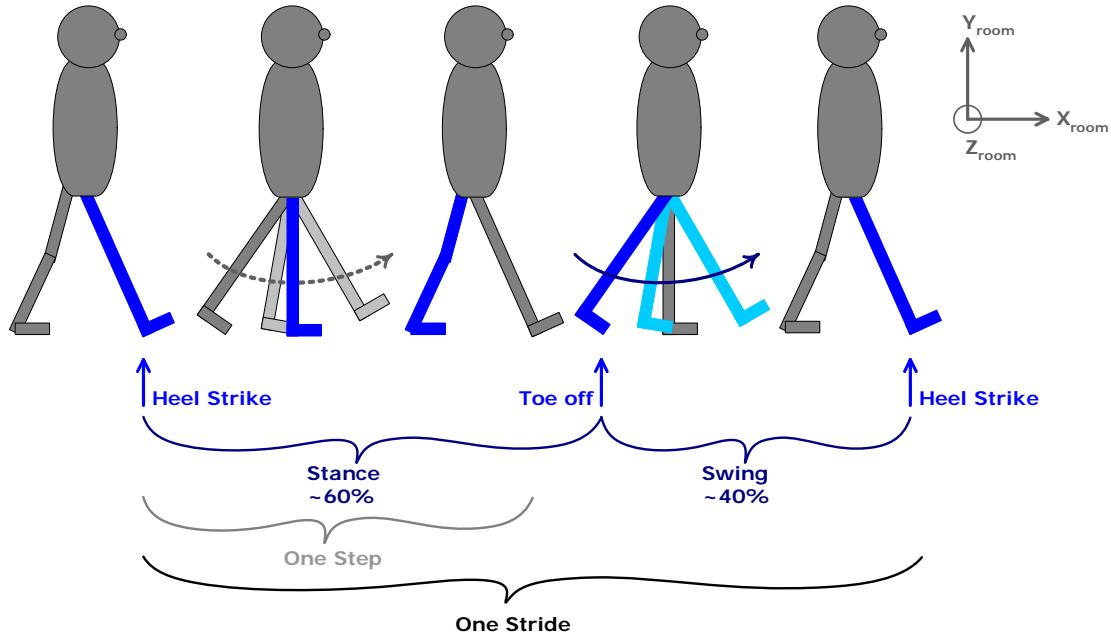


Figure A.1 The Gait Cycle

A.2 Parkinson's Disease

Parkinson's disease (PD) is a chronic and progressive movement disorder resulting from the loss of dopamine-producing neurons in the *substantia nigra* area of the brain. Dopamine is a chemical messenger which relays neurological signals for the coordination and controlled initiation of movement. With the loss of dopamine-producing neurons, dopamine levels fall, resulting in the symptoms of PD. The diagnosis of PD is made following a physical exam for common PD symptoms, and ruling out other conditions with similar

symptoms. The exact cause of PD is not yet known, though there appears to be some genetic contribution. Environmental toxins may play a role as well.

The most common symptoms of PD are tremors of the limbs, jaw, and face, rigidity of the limbs and trunk, bradykinesia (slowed movement), and postural instability, resulting in impaired balance and coordination. Manifestation of these symptoms in gait result in short steps and a shuffling gait, called festination, in difficulty initiating gait, called freezing, in difficulty to turn, and in loss of balance.

Parkinson's disease is generally treated first with medications designed either to work by increasing dopamine levels (e.g. by providing precursors, or by activating the release of stored dopamine), or by activating the dopamine receptor directly; medications to slow progression of the disease are in development. Surgical interventions are available, generally to patients who are not satisfied with the results of medication-controlled treatment. Pallidotomy (to alleviate rigidity and bradykinesia) and thalamotomy (to alleviate tremors) are procedures in which small regions of the brain are permanently destroyed to alleviate symptoms. Deep brain stimulation implants, considered by some to be a safer and more effective surgical treatment, involve an electrode implanted in the brain to provide an electrical impulse to a targeted region to alleviate symptoms (the electrodes are connected via wires to an impulse generator placed under the subject's clavicle) [118] [119].

Appendix B

SUBJECT TESTING

This appendix describes the subject testing used to acquire data used in Chapter 5 for validation and in Chapter 6 for pattern recognition. The study design, including subject recruitment, consent forms, and the testing protocol, is discussed, and information about the subjects is included as well. Donna Moxley Scarborough, MS/PT, was the principal tester at the Massachusetts General Hospital (MGH) Biomotion Laboratory (BML).

B.1 Study Design

The subject testing involved placing the GaitShoe instrumentation on the subjects' own walking shoes, with the insole inside the shoe and the shoe attachment mounted to the posterior aspect of each shoe. The small antenna on the circuit board transmitted the sensor information to the receiving transmitter. All signals were digitized and saved on a laptop computer set within 30 feet of the shoes during data collection. The subject underwent simultaneous gait evaluation using the MGH Biomotion Laboratory's Selspot II data acquisition system. Each subject was asked to perform a series of locomotor tasks, while both gait evaluation systems simultaneously collected data. The gait parameters collected from the two systems was analyzed and compared to validate the analysis of gait parameters from the data acquired by the GaitShoe, as discussed in Chapter 5; in addition, the data were used for the pattern recognition study, as discussed in Chapter 6. The informa-

tion gathered did not offer any direct benefits to the subjects tested. After the subjects complete the protocol, they concluded their participation in the study.

B.1.1 Subject Recruitment

The subjects with healthy gait were recruited via e-mail and word of mouth. When subjects replied with interest in the study, they were provided with more details about the study, and an appointment was set up at the Biomotion Lab.

The subjects with Parkinson's disease were recruited by collaborators Drs. Stephen Parker and Leslie Shinobu of the MGH Department of Neurology, who performed initial screening of PD patients within their practice and described the research project to prospective subjects. If a subject verbally agreed to being contacted by phone from a Biomotion Laboratory study representative, Drs. Parker or Shinobu provided the subject's telephone number to the study's administrator who contacted prospective subjects via telephone to provide more details about the research project. With further agreement from the subject, an appointment was set up at the Biomotion Lab.

All subjects were adults who could understand and follow basic directions. Persons were excluded if they reported acute pain which prevented performance of their comfortable, typical movement. Similarly, persons were excluded if they have a unstable medical condition such as hypertension or diabetes mellitus.

B.1.2 Consent Forms

The protocols for this study was approved by both the MGH Institutional Review Board (IRB) and the MIT Committee On the Use of Humans as Experimental Subjects (COUHES). The subjects all consented in accordance with the MGH IRB and the MIT COUHES. The MGH IRB consent form for subjects with healthy gait is shown in Figure B.1, the MGH IRB consent form for subjects with difficulty walking is shown in Figure B.2, and the MIT COUHES consent form is shown in Figure B.3 (the two MGH forms were stamped with the IRB approval; the stamp is not visible in these figures). Subject testing

took place from the March 25, 2003 through June 6, 2003; the Health Insurance Portability and Accountability Act of 1996 (HIPAA) went into effect April 15, 2003, so the final eleven subjects (tested after April 15, 2003) additionally signed the HIPAA paperwork.

Figure B.1 MGH consent form for subjects with healthy gait

Figure B.2 MGH consent form for subjects with difficulty walking

Research Consent Form
Massachusetts Institute of Technology

Imprint Patient ID Number

STUDY CONTACTS

The principal investigator for this study at MIT is Professor Joseph Paradiso. He can be reached at (617) 253-6215.

The principal investigator for this study at MGH is Donna Scarborough. She can be reached at (617) 726-3406 (weekdays 9am-4pm).

REQUEST FOR MORE INFORMATION

You may ask more questions about the study at any time. The investigator(s) will provide their telephone number so that they are available to answer your questions or concerns about the study. You will be informed of any significant new findings discovered during the course of this study that might influence your continued participation.

If during the study or later, you wish to discuss your rights as a research subject, your participation in the study and/or concerns about the study, a research-related injury with someone not directly involved in the study, or if you feel under any pressure to enroll in this study or to continue to participate in this study, you are asked to contact the Committee on the Use of Humans as Experimental Subjects at MIT at (617) 253-6787, or the Human Research Committees at BWH at (617) 732-7200, at MGH at (617) 726-3493, or at the Protocol Administration Office at DFCI at (617) 632-3029.

Figure B.3 MIT COUHES Consent Form

<p>GAIT TESTING PROTOCOL FOR MGH BIOMOTION LAB <i>Healthy and Pathological Gait Subjects</i> <i>Show gait evaluator testing</i></p> <p>Name _____ ID# _____ Test Date _____</p> <p>Home address, phone _____</p> <p>Diagnosis _____ DOB _____</p> <p>Sex M F Height _____ in Weight _____ lb</p> <p>Activities/ vocational/ recreational? _____</p> <p>Use cane/crutch? Y N If not, when last used >50% of a day? _____ / _____</p> <p>Receiving PT now? Y N When last received? _____ / _____</p> <p>History of Falls? _____ in the past year _____ in the past month (Fall=inadvertently coming to ground)</p> <p>Other medical hx: Vascular/Ortho (Record ONLY if pt. volunteers; do not inquire otherwise)</p> <p>***** Above to be completed at time of subject scheduling *****</p> <p>Hospitalizations? Record ONLY if pt. volunteers; do not inquire otherwise</p> <p>Health information provided from subject to tester during setup: (Cardiovascular system, signs of distress)</p> <p>Comments: _____ _____ _____ _____</p> <p>Signed informed consent? Signed HIPAA authorization?</p>	<p>SUBJECT NAME _____ Visit Dates 1: _____ 2: _____</p> <p>1. Measure height and circumferences 2. Which hand do you eat with? _____ Right _____ Left _____?</p> <p>3. Don arrays and locate, take photographs front/back side during erect standing. Track lights only on, except as noted.</p> <p>4. Palpate and locate anatomical markers at heel, great toe, and first and fifth metatarsal heads. Mark and transfer location to paper.</p> <p>5. Perform practice trial for all tasks except when identified with D – Subject gets NO practice trial. D = Demonstrate on helper or self.</p> <p>Testing conditions with lights off except track lights</p> <p>Time Completed Activities _____ NOTE SOME REPEATED X2 EACH VISIT</p> <p>I. Standing: T Look straight ahead, stand as still as possible; arms folded across your chest, grasping your elbows. Place your feet so they point forward [and parallel to X axis]* ? See samples</p> <p>CHECK VISIT# 1 2 (note gait is WITH arm swing)</p> <p>_____ 1. Gait initiation speed #1/2. Affected limb then next run non-affected on FP if possible. "Move forward in as straight a line as possible, walking at your normal pace, as if you were taking a brisk walk in the park. Begin after I say 1-2-ready-go." * see samples, (WD)</p> <p>_____ 2. Static standing (feet (middle) 30 cm (12") apart#2 and feet parallel (+Joint Centers only if array movement during task) * see samples, (WD)</p> <p>_____ 3. Calibration trials Dorsiflex range, reverse: _____ Foot rotation: _____ Foot rotation, reverse: _____ Foot roll: _____ Leg roll: _____ Leg swing: _____</p> <p>_____ 4. Chair stand #1/2. Feet on FP, 18" H legs, feet to 10° apart and parallel, backrests, sit from seat edge 100% (one foot to MTP height). Back with hands grasping elbows, elbows against chest. "Get up from the chair looking at the dot for the way you usually do, after I say 1-2-ready-go. Begin rising on "go" After you have stood, stand as still as possible until I say stop. 1-2-READY-GO." * see samples, (WD)</p> <p>_____ 5. Gait initiation free speed #1/2. Start at FP, lead w/ affected leg winging first. (Due data collection prior to command to begin gait)</p> <p>_____ 6. Gait free speed #1/2. Affected limb then next run non-affected on FP if possible. "Move forward in as straight a line as possible looking at the dot, walking at your normal pace, as if you were taking a brisk walk in the park. Begin after I say 1-2-ready-go. After each trial, record start and final number." * see samples, (WD)</p> <p>_____ 7. Gait: Disturb recall #1 (Disturbance of static memory)</p> <p>_____ 8. Gait: Disturb creative #1 (favorite season and why)</p> <p>_____ 9. Gait: Disturb motor #1 (self repeated catching of ball)</p> <p>_____ 10. Gait: Wall walk #1 (counting backwards from random #) "While walking whisper subtracting serial 7's from the number I give you after I say 'go' (e.g., 143, 7 must be 100-999). After each trial, record start and final number: (WD) # _____</p> <p>_____ 11. Gait: paced 120 BPM #2/2. Same as #16 except: "Walk to this beat"(WD)</p> <p>_____ 12. Gait: paced distillation #1. Same as #11 except: "Walk to this beat WHILE you are silently subtracting serial 7's starting with the number I give you after I say go (e.g., 143, 7 must be 100-999)." After each trial, record start and final number (WD)</p> <p>_____ 13. Gait: visual cue (walk with spaced horizontal line cues on floor) #1/2 "Walk placing each foot to land on the consecutive line" (WD).</p> <p>_____ 14. Gait: visual cue (paced cued combined #1/2 "Walk placing each foot on the next consecutive line, with each step breaking the beat of the metronome"</p> <p>_____ 15. Gait: visual distractor (walk with lines on floor disrupted) #1/2 "Walk in a straight manner as if you were taking a brisk walk in the park"</p> <p>_____ 16. Gait: wall walk #1 (methodology) 30 cm (12") apart#2 and feet parallel</p> <p>_____ 17. Turn around Can be with Soplex disconnected and FP data only # 1/2 "Walk forward in a straight line and turn around just before the hanging tape measure and walk back where you started."</p> <p>*****</p> <p>IF time and subject able:</p> <p>18 Step 100 BPM #1/2 "Go up and down this 3' step with your arms swinging naturally, non affected up then affected up then non affected leg down then affected down;" 2+4 steps before data collection. 30 sec (N,D,WD). If unable, step 100 BPM #1/2 Same as above, except 10 s @ 100 bpm, 10 s @ 80 and 10 s @ 100. "I will change the beat at some point. Please respond to the change as quickly as possible." 30 sec samples (N,D,WD) If unable, step in place.</p> <p>Tester Name _____</p>
--	--

Figure B.4 Subject Testing Protocol, page 1 (left) and page 2 (right)

B.1.3 Protocol

The protocol followed during subject testing is shown above in Figure B.4.

The subjects were asked to walk in a variety of ways. First, each subject was first asked to walk at his or her own pace (termed "free gait" for use in Chapter 6). Next, a number of calibration routines were carried out, including "chair rise", where the subject stood from a seated position. Following the calibrations, the subject started gait while within the viewing volume of the BML cameras, to collect data with the BML system about the initiation of gait, and then one to three "free gait" trials were carried out. In each of the following four trials, the subject was told to do a task (detailed in Figure B.3) designed to provide distraction ("distracted gait" in Chapter 6). Next, the metronome was turned on to 120 beats per minute, for two to three trials ("paced gait" in Chapter 6). Finally, lines were placed on the floor with a separation of approximately 1 m, and the subject was asked to step on the lines while walking (included with the "distracted gait" group in Chapter 6). Collection from the BML optical system concluded with "static standing", while the subject stood still with the feet 30 cm apart (this data was used to determine the orientations of the accelerometers with respect to the horizontal, for each subject). The testing concluded with data collected from the BML force plate and the GaitShoe while the subject walked forward to the center of the viewing volume, turned, and walked back to the starting point; this data was not analyzed for this thesis, but was collected for future work in analyzing gait which includes turns.

B.2 Subject Information

A total of sixteen subjects were recruited for the validation of the GaitShoe; they were provided with identification numbers¹ consecutively from 11 to 26. Gender, age, height,

1. An additional ten subjects (01 to 10) were recruited for prototype testing and evaluation of the GaitShoe; the data from these initial subjects was not used in the final analysis presented in this thesis.

weight, and presence of Parkinson's disease are detailed in Table B.1 (in Chapter 6, the subjects are referred to by a coded name).

TABLE B.1 Information about volunteers for the subject testing

ID	Gender [Male/Female]	Age [years]	Height [m]	Weight [kg]	Parkinson's disease [Yes/No]
11	F	24.9	1.6	48.2	N
12	F	28.2	1.6	59.1	N
13	M	25.3	1.8	75.0	N
14	M	27.5	1.8	115.0	N
15	F	48.2	1.6	52.3	N
16	F	28.6	1.6	50.0	N
17	F	54.0	1.7	54.5	N
18	F	26.8	1.6	66.4	N
19	F	27.4	1.7	55.9	N
20	F	53.8	1.6	63.6	Y
21	F	26.9	1.7	69.5	N
22	F	65.9	1.7	68.2	Y
23	M	64.9	1.8	94.5	Y
24	M	76.4	1.7	77.7	Y
25	F	65.4	1.6	52.3	Y
26	M	30.3	1.8	90.9	N

The data collected for Subject 21 were excluded from the analysis because the antennas on both the left and right GaitShoe attachments were in need of repair; this was not discovered until after data testing was underway, and resulted in very poor data transmission, with a data collection rate from the GaitShoe lower than 50 Hz for each foot, rather than the usual 75 Hz.

Thus, the final cohort of subjects included five subjects with Parkinson's disease (2 males and 3 females), and ten subjects with normal gait¹ (7 females and 3 females).

1. One of these ten subjects was diagnosed with *myasthenia gravis* (a neurological disease usually affecting face muscles), however, no changes in gait were observed by the physical therapists at the BML, or in GaitShoe data used for the pattern recognition in Chapter 6. This subject was therefore not uniquely labeled, but included as a subject with normal gait.

Appendix C

PATTERN RECOGNITION INFORMATION

This appendix accompanies Chapter 6, and explains terminology commonly used in pattern recognition (see also Pattern Classification by Duda, Hart, and Stork [96]). This appendix also contains the complete results from the classification. Results are presented in the same order as in Chapter 6; many of the tables here were included in Chapter 6 as well (results from the additional neural net studies are only in Section 6.6.5).

C.1 Terminology

Bayes Decision Theory: See Appendix 6.1.2.

CART: Classification and Regression Trees (a type of decision tree); see Section 6.1.1. Figure C.1 shows a sample tree, with three classes, 123 total samples, and using four features to create the tree.

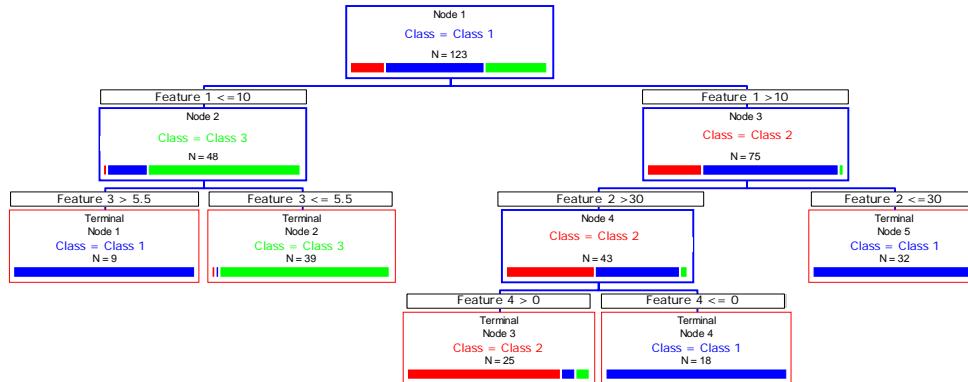


Figure C.1 Sample CART tree

Class: Descriptor for a group of *samples*, e.g. "apple" or "banana" or "orange".

Confusion Matrix: Table of results showing how the *sample* in the *testing set* were classified. Table C.1 shows a sample *confusion matrix* for a classification of apples, bananas, and oranges. Each row shows the results for a single *class*; in this example, 48 apples were correctly classified as apples, and 2 apples were misclassified as oranges, for a 96.0% classification rate for the apples. Similarly, all 50 bananas (100%) were classified correctly, and 40 oranges (80%) were classified correctly.

TABLE C.1 Sample results presented in a confusion matrix

	Apples	Bananas	Oranges	% correct
Apples	48	0	2	96.0
Bananas	0	50	0	100.0
Oranges	10	0	40	80.0
Overall:				92.0

The overall classification rate is calculated by the sum of the correct classifications (the numbers along the diagonal), divided by the total number of *samples* in the *testing set*; in this example:

$$\text{Overall Classification Rate} = \frac{48 + 50 + 40}{50 + 50 + 50} = 92.0\%. \quad (8.1)$$

Data Set: Collection of *samples*, including samples from all classes. The *data set* is typically split into a *training set* and a *testing set*.

Features: Continuous or categorical properties that (ideally) distinguish between the *classes*. For classifying apples, bananas, and oranges, categorical features might be color or shape, continuous features might be hue or mass. A feature such as "food type" would not be useful, as all three classes are types of fruit.

Leave One Out: A method of evaluating the classifier by having N *training* and *testing sets*, where N is the total number of *samples*, and each *testing set* consists of a single sample and its corresponding *training set* consists the N-1 *samples* not in the *testing set*.

Neural Networks: See Section 6.1.4.

Sample: A single piece of data, consisting of a class label and the features.

SVMs: Support Vector Machines; see Section 6.1.3.

Testing Set: Group of *samples* used to "test" the classifier; the classifier is applied to these samples to see what percentage of the testing set is correctly classified. If the *testing set* is used to adjust the parameters of the classifier (as it is with the CART software), the results are termed "overly optimistic;" it is good practice to use a separate "evaluation set" to avoid this.

Training Set: Group of *samples* used to "train" the classifier; these samples are used to determine the parameters of the classifier to distinguish between the *classes* of the samples within the training set.

C.2 Complete Results

C.2.1 Hypothesis 1

Hypothesis 1 compared group NLFY (females with normal gait, younger than 30 years old), with group NLF (females with normal gait, older than 48 years old), and group PDF (females with Parkinson's disease, older than 54 years old). The training and testing groups are described in Section 6.5.

The results for Hypothesis 1, using modified leave one out, are shown in Tables C.2, C.3, and C.4.

TABLE C.2 CART Hypothesis 1 results, using modified leave one out

	NLFY	NLF	PDF	% correct	% NLFY or NLF
NLFY	122	10	3	90.4	97.8
NLF	6	47	1	87.0	98.2
PDF	0	3	78	96.3	3.7
Overall:				91.5	

TABLE C.3 Naïve Bayes Hypothesis 1 results, using modified leave one out

	NLFY	NLF	PDF	% correct	% NLFY or NLF
NLFY	129	2	4	95.6	97.0
NLF	2	50	2	92.6	96.3
PDF	0	3	78	96.3	3.7
Overall:				95.2	

TABLE C.4 Neural Net Hypothesis 1 results, using modified leave one out

	NLFY	NLF	PDF	% correct	% NLFY or NLF
NLFY	131	2	2	97.0	98.5
NLF	3	50	1	92.6	98.2
PDF	0	0	81	100.0	0.0
Overall:				97.0	

The results for Hypothesis 1, using "leave one subject out", are shown summed in Table C.5 and by subject in Table C.6.

TABLE C.5 CART Hypothesis 1 results, by class, using leave one subject out

	NLFY	NLF	PDF	% correct	% NLFY or NLF
NLFY	66	21	5	71.7	94.6
NLF	32	4	1	10.8	97.3
PDF	4	13	47	73.4	26.7
Overall:				60.6	

TABLE C.6 CART Hypothesis 1 results, by subject, leave one subject out

	NLFY	NLF	PDF	% correct	% NLFY or NLF
NLFY-A	16	0	0	100.0	100.0
NLFY-B	8	9	0	47.1	100.0
NLFY-C	10	9	2	47.6	90.5
NLFY-D	18	0	0	100.0	100.0
NLFY-E	14	3	3	70.0	85.0
NLF-A	15	4	0	21.1	100.0
NLF-B	17	0	1	0.0	94.4
PDF-A	0	0	20	100.0	0.0
PDF-B	0	0	23	100.0	0.0
PDF-C	4	13	4	19.1	81.0
Overall:				60.6	

The results for Hypothesis 1, using "free gait", "distracted gait", or "paced gait" (gait types are described in Section B.1) as the test set, are shown in Tables C.7, C.8, and C.9.

TABLE C.7 CART Hypothesis 1 results, using "free gait" as the test set

	NLFY	NLF	PDF	% correct	% NLFY or NLF
NLFY	29	2	7	76.3	81.6
NLF	4	8	1	61.5	92.3
PDF	1	2	18	85.7	14.3
Overall:				76.4	

TABLE C.8 CART Hypothesis 1 results, using "distracted gait" as the test set

	NLFY	NLF	PDF	% correct	% NLFY or NLF
NLFY	27	0	3	90.0	90.0
NLF	1	14	0	93.3	100.0
PDF	1	0	24	96.0	4.0
Overall:				92.9	

TABLE C.9 CART Hypothesis 1 results, using "paced gait" as the test set

	NLFY	NLF	PDF	% correct	% NLFY or NLF
NLFY	19	2	3	79.2	87.5
NLF	0	9	0	100.0	100.0
PDF	0	0	18	100.0	0.0
Overall:				90.2	

As discussed in Chapter 6, SVMs were used to classify between two classes. Hypothesis 1.1 compared NLFY with NLF, Hypothesis 1.2 compared NLF with PDF, and Hypothesis 1.3 compared NLFY with PDF. The results for each, using modified leave one out, are shown in Tables C.10, C.11, and C.12.

TABLE C.10 SVM Hypothesis 1.1 results, using modified leave one out

	NLFY	NLF	% correct
NLFY	130	5	96.3
NLF	4	50	92.6
Overall:			95.2

TABLE C.11 SVM Hypothesis 1.2 results, using modified leave one out

	NLF	PDF	% correct
NLF	46	8	85.2
PDF	1	80	98.8
Overall:			93.3

TABLE C.12 SVM Hypothesis 1.3 results, using modified leave one out

	NLFY	PDF	% correct
NLFY	132	3	97.8
PDF	3	78	96.3
Overall:			97.2

The results for Hypotheses 1.1, 1.2, and 1.3, using "leave one subject out", are shown in Tables C.13, C.14, and C.15.

TABLE C.13 SVM Hypothesis 1.1 results, using leave one subject out

	NLFY	NLF	% correct
NLFY	80	12	87.0
NLF	34	3	8.1
Overall:			64.3

TABLE C.14 SVM Hypothesis 1.2 results, using leave one subject out

	NLF	PDF	% correct
NLF	10	27	27.0
PDF	32	32	50.0
Overall:			41.6

TABLE C.15 SVM Hypothesis 1.3 results, using leave one subject out

	NLFY	PDF	% correct
NLFY	89	3	96.7
PDF	28	36	56.3
Overall:			80.1

The results for Hypotheses 1.1, 1.2, and 1.3, using "free gait" as the test set, are shown in Tables C.16, C.17, and C.18.

TABLE C.16 SVM Hypothesis 1.1 results, using "free gait" as the test set

	NLFY	NLF	% correct
NLFY	38	0	100.0
NLF	3	10	76.9
Overall:			94.1

TABLE C.17 SVM Hypothesis 1.2 results, using "free gait" as the test set

	NLF	PDF	% correct
NLF	12	1	92.3
PDF	4	17	81.0
Overall:			85.3

TABLE C.18 SVM Hypothesis 1.3 results, using "free gait" as the test set

	NLFY	PDF	% correct
NLFY	37	1	97.4
PDF	1	20	95.2
Overall:			96.6

The results for Hypotheses 1.1, 1.2, and 1.3, using "distracted gait" as the test set, are shown in Tables C.19, C.20, and C.21.

TABLE C.19 SVM Hypothesis 1.1 results, using "distracted gait" as the test set

	NLFY	NLF	% correct
NLFY	29	1	96.7
NLF	1	14	93.3
Overall:			95.6

TABLE C.20 SVM Hypothesis 1.2 results, using "distracted gait" as the test set

	NLF	PDF	% correct
NLF	12	3	80.0
PDF	4	21	84.0
Overall:			82.5

TABLE C.21 SVM Hypothesis 1.3 results, using "distracted gait" as the test set

	NLFY	PDF	% correct
NLFY	30	0	100.0
PDF	3	22	88.0
Overall:			94.5

The results for Hypotheses 1.1, 1.2, and 1.3, using "paced gait" as the test set, are shown in Tables C.22, C.23, and C.24.

TABLE C.22 SVM Hypothesis 1.1 results, using "paced gait" as the test set

	NLFY	NLF	% correct
NLFY	22	2	91.7
NLF	1	8	88.9
Overall:			90.9

TABLE C.23 SVM Hypothesis 1.2 results, using "paced gait" as the test set

	NLF	PDF	% correct
NLF	8	1	88.9
PDF	0	18	100.0
Overall:			96.3

TABLE C.24 SVM Hypothesis 1.3 results, using "paced gait" as the test set

	NLFY	PDF	% correct
NLFY	21	3	87.5
PDF	0	18	100.0
Overall:			92.9

C.2.2 Hypothesis 2

Hypothesis 2 compared the combined group of NLFY and NLF (resulting in a group of all females with normal gait), with group PDF (all females with Parkinson's disease).

The results for Hypothesis 2, using modified leave one out, are shown in Tables C.25, C.26, C.27, and C.28.

TABLE C.25 CART Hypothesis 2 results, using modified leave one out

	NLF/NLFY	PDF	% correct
NLF/NLFY	181	8	95.8
PDF	1	80	98.8
Overall:			96.7

TABLE C.26 SVM Hypothesis 2 results, using modified leave one out

	NLF/NLFY	PDF	% correct
NLF/NLFY	181	8	95.8
PDF	2	79	97.5
Overall:			96.3

TABLE C.27 Naïve Bayes Hypothesis 2 results, using modified leave one out

	NLF/NLFY	PDF	% correct
NLF/NLFY	181	8	95.8
PDF	4	77	95.1
Overall:			95.6

TABLE C.28 Neural Net Hypothesis 2 results, using modified leave one out

	NLF/NLFY	PDF	% correct
NLF/NLFY	187	2	98.9
PDF	0	81	100.0
Overall:			99.3

The results for Hypothesis 2, using "leave one subject out", are shown summed in Tables C.29, C.31, and C.33, and by subject in Tables C.30, C.32, and C.34.

TABLE C.29 CART Hypothesis 2 results, using leave one subject out

	NLF/NLFY	PDF	% correct
NLF/NLFY	123	6	95.3
PDF	21	43	67.2
Overall:			86.0

TABLE C.30 CART Hypothesis 2 results, by subject, using leave one subject out

	NLF/NLFY	PDF	% correct
NLFY-A	16	0	100.0
NLFY-B	17	0	100.0
NLFY-C	19	2	90.5
NLFY-D	18	0	100.0
NLFY-E	17	3	85.0
NLF-A	19	0	100.0
NLF-B	17	1	94.4
PDF-A	4	16	80.0
PDF-B	0	23	100.0
PDF-C	17	4	19.0
Overall:			86.0

TABLE C.31 SVM Hypothesis 2 results, using leave one subject out

	NLF/NLFY	PDF	% correct
NLF/NLFY	114	15	88.4
PDF	33	31	48.4
Overall:			75.1

TABLE C.32 SVM Hypothesis 2 results, by subject, using leave one subject out

	NLF/NLFY	PDF	% correct
NLFY-A	15	1	93.8
NLFY-B	17	0	100.0
NLFY-C	21	0	100.0
NLFY-D	18	0	100.0
NLFY-E	20	0	100.0
NLF-A	16	3	84.2
NLF-B	7	11	38.9
PDF-A	6	14	70.0
PDF-B	12	11	47.8
PDF-C	15	6	28.6
Overall:		75.1	

TABLE C.33 Neural Net Hypothesis 2 results, using leave one subject out

	NLF/NLFY	PDF	% correct
NLF/NLFY	122	7	94.6
PDF	18	46	71.9
Overall:			87.0

TABLE C.34 Neural Net Hypothesis 2 results, by subject, using leave one subject out

	NLF/NLFY	PDF	% correct
NLFY-A	16	0	100.0
NLFY-B	17	0	100.0
NLFY-C	20	1	95.2
NLFY-D	18	0	100.0
NLFY-E	15	5	75.0
NLF-A	19	0	100.0
NLF-B	17	1	94.4
PDF-A	5	15	75.0
PDF-B	0	23	100.0
PDF-C	13	8	38.1
Overall:			87.0

The results for Hypothesis 2, using "free gait", "distracted gait", or "paced gait" as the test set, are shown in Tables C.35, C.36, C.37, C.38, C.39, and C.40.

TABLE C.35 CART Hypothesis 2 results, using "free gait" as the test set

	NLF/NLFY	PDF	% correct
NLF/NLFY	43	8	84.3
PDF	2	19	90.5
Overall:			86.1

TABLE C.36 SVM Hypothesis 2 results, using "free gait" as the test set

	NLF/NLFY	PDF	% correct
NLF/NLFY	51	0	100.0
PDF	2	19	90.5
Overall:			97.2

TABLE C.37 CART Hypothesis 2 results, using "distracted gait" as the test set

	NLF/NLFY	PDF	% correct
NLF/NLFY	38	7	84.4
PDF	1	24	96.0
Overall:			88.6

TABLE C.38 SVM Hypothesis 2 results, using "distracted gait" as the test set

	NLF/NLFY	PDF	% correct
NLF/NLFY	44	1	97.8
PDF	3	22	88.0
Overall:			94.3

TABLE C.39 CART Hypothesis 2 results, using "paced gait" as the test set

	NLF/NLFY	PDF	% correct
NLF/NLFY	31	2	93.9
PDF	1	17	94.4
Overall:			94.1

TABLE C.40 SVM Hypothesis 2 results, using "paced gait" as the test set

	NLF/NLFY	PDF	% correct
NLF/NLFY	30	3	90.9
PDF	0	18	100.0
Overall:		94.1	

C.2.3 Hypothesis 3

Hypothesis 3 classified each of the ten subjects as individuals; the results, using cross-validation, are shown in Tables C.41, C.42, and C.43.

TABLE C.41 Neural Network Hypothesis 3 results, by subject, using cross-validation

	NLFY A	NLFY B	NLFY C	NLFY D	NLFY E	NLF A	NLF B	PDF A	PDF B	PDF C	% correct	% NLFY or NLF
NLFY-A	15	0	0	1	0	0	0	0	0	0	93.8	100.0
NLFY-B	0	17	0	0	0	0	0	0	0	0	100.0	100.0
NLFY-C	0	0	20	0	1	0	0	0	0	0	95.2	100.0
NLFY-D	0	0	0	18	0	0	0	0	0	0	100.0	100.0
NLFY-E	0	0	0	0	18	1	0	0	0	1	90.0	95.0
NLF-A	0	0	0	0	0	19	0	0	0	0	100.0	100.0
NLF-B	0	0	0	0	0	0	18	0	0	0	100.0	100.0
PDF-A	0	0	0	0	0	0	0	20	0	0	100.0	0.0
PDF-B	0	0	0	0	0	0	0	0	23	0	100.0	0.0
PDF-C	0	0	0	0	0	0	0	1	0	20	95.2	0.0
Overall:										97.4		

TABLE C.42 CART Hypothesis 3 results, by subject, using cross-validation

	NLFY A	NLFY B	NLFY C	NLFY D	NLFY E	NLF A	NLF B	PDF A	PDF B	PDF C	% correct	% NLFY or NLF
NLFY-A	14	1	0	1	0	0	0	0	0	0	87.5	100.0
NLFY-B	0	17	0	0	0	0	0	0	0	0	100.0	100.0
NLFY-C	0	0	16	0	4	1	0	0	0	0	76.2	100.0
NLFY-D	0	1	0	16	0	0	1	0	0	0	88.9	100.0
NLFY-E	0	0	4	0	13	3	0	0	0	0	65.0	100.0
NLF-A	0	0	2	0	3	13	1	0	0	0	68.4	100.0
NLF-B	0	2	0	0	1	0	12	0	0	3	66.7	83.3
PDF-A	0	0	0	0	2	0	0	17	1	0	85.0	10.0
PDF-B	0	0	0	0	0	0	0	2	21	0	91.3	0.0
PDF-C	0	0	1	0	0	0	1	0	0	19	90.5	9.5
Overall:											81.9	

TABLE C.43 Naïve Bayes Hypothesis 3 results, by subject, using cross-validation

	NLFY A	NLFY B	NLFY C	NLFY D	NLFY E	NLF A	NLF B	PDF A	PDF B	PDF C	% correct	% NLFY or NLF
NLFY-A	15	0	0	1	0	0	0	0	0	0	93.8	100.0
NLFY-B	0	17	0	0	0	0	0	0	0	0	100.0	100.0
NLFY-C	0	0	18	0	3	0	0	0	0	0	85.7	100.0
NLFY-D	0	0	0	17	1	0	0	0	0	0	94.4	100.0
NLFY-E	0	0	3	0	16	0	0	0	0	1	80.0	95.0
NLF-A	0	0	1	0	0	18	0	0	0	0	94.7	100.0
NLF-B	0	0	0	0	2	0	16	0	0	0	88.9	100.0
PDF-A	0	0	1	0	0	0	0	19	0	0	95.0	5.0
PDF-B	0	0	1	0	0	0	0	0	22	0	95.7	4.3
PDF-C	0	0	0	0	0	0	1	1	0	0	90.5	4.8
Overall:											91.7	

Appendix D

HARDWARE INFORMATION

This appendix contains information required for the building of the GaitShoe circuit boards and other hardware, and contains part numbers and purchasing information for all parts used.

D.1 Schematics and Board Layouts

D.1.1 Inertial Measurement Unit (IMU) Board, Rev. 5

The schematic for the IMU board (Rev. 5) is shown in Figure D.1. The board layouts with part placement information are shown in Figure D.2 (top side) and Figure D.3 (bottom side).

In Figure D.2, the two footprints for the two different versions of the Murata gyroscopes are visible. As discussed in Section 3.3.2, the ENC-03J was used in all testing, but the footprint for the surface mount ENC-03M was also included on the IMU board (the alternate pin connections can be seen in Figure D.1) in case the ENC-03J becomes no longer available.

Similarly, the Analog Devices gyroscope (the ADXRS150) was a demo DIP version and is not available commercially. It is now available in a 32-pin ball grid array surface-mount package; pin-to-pin mappings between this package and the layout on the board (e.g. the sixteen pin header would plug into the holes for the current DIP) are shown in Figure D.4.

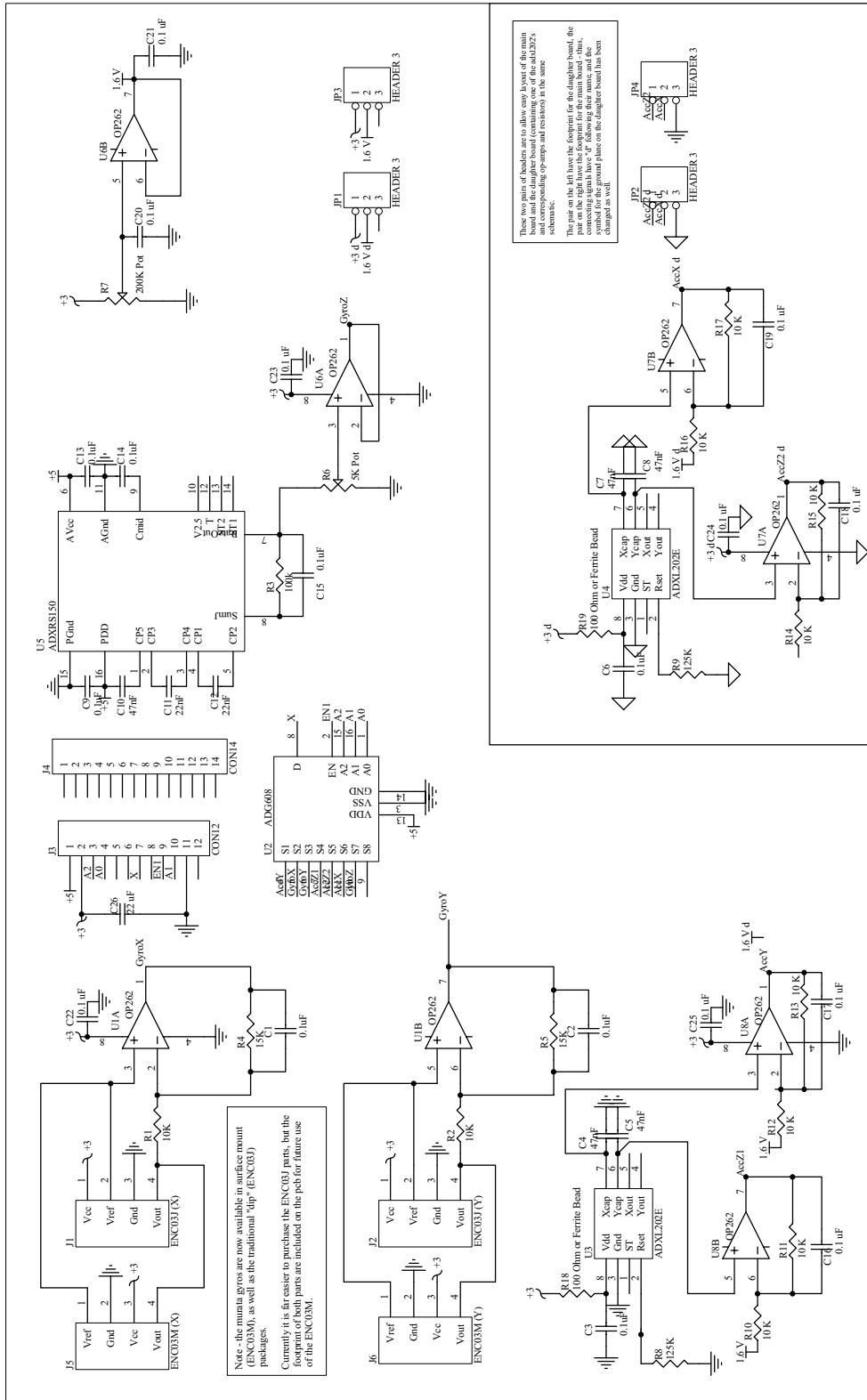


Figure D.1 Schematic of the IMU board

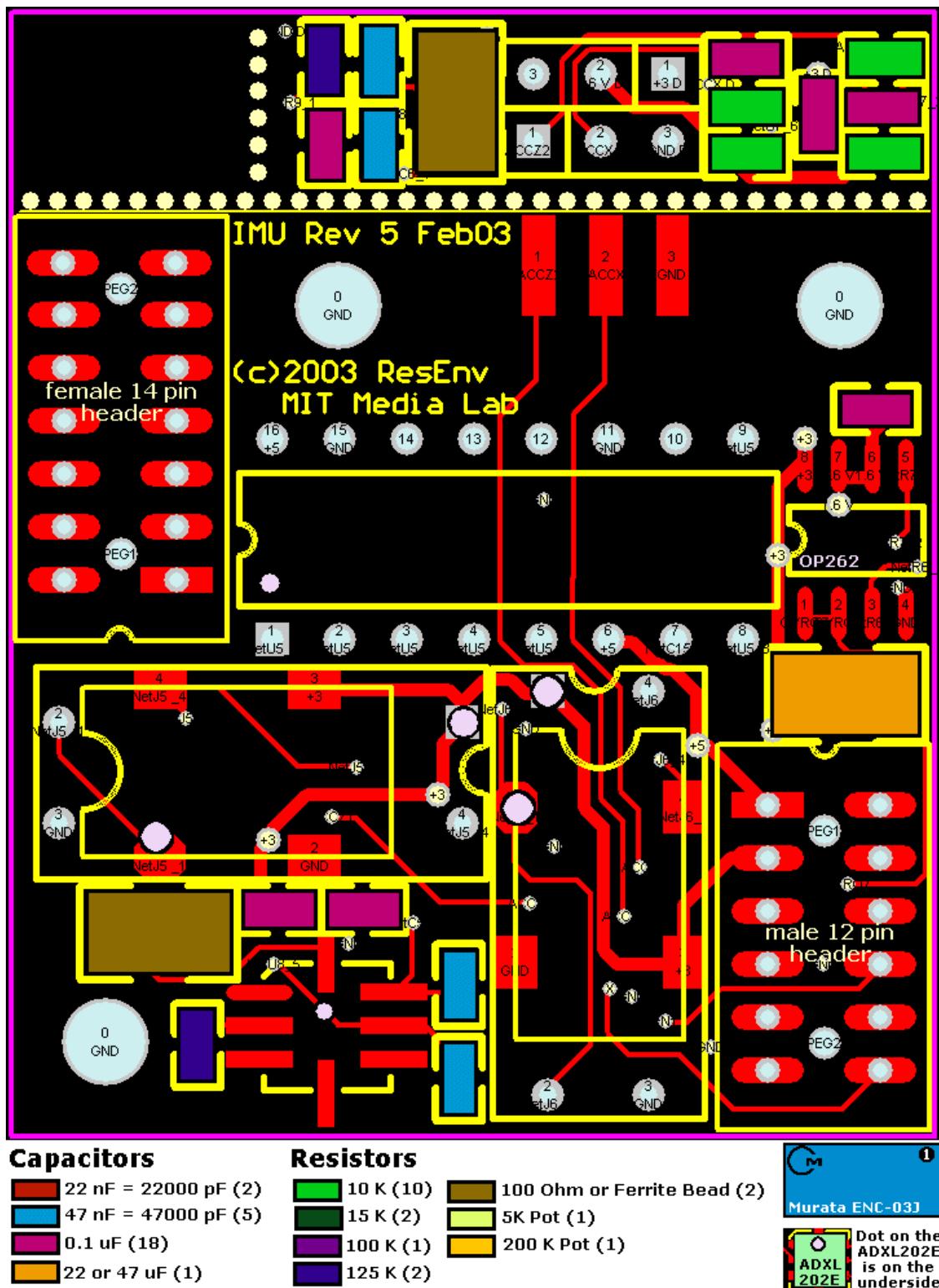


Figure D.2 Layout of the top side of the IMU board

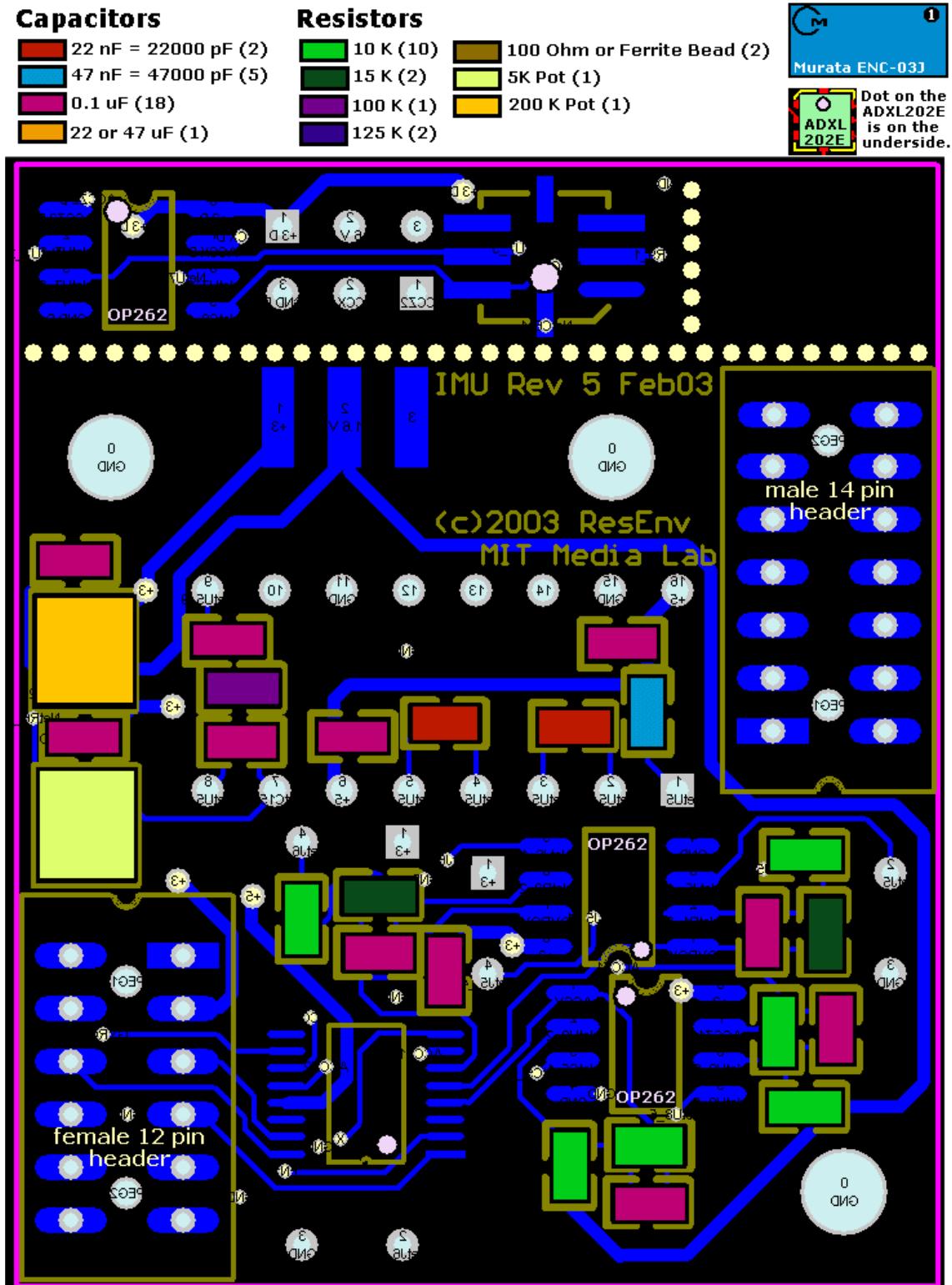


Figure D.3 Layout of the bottom side of the IMU board

As indicated, the capacitors should be included on the daughter board, to be located as close to the ADXRS150 as possible.

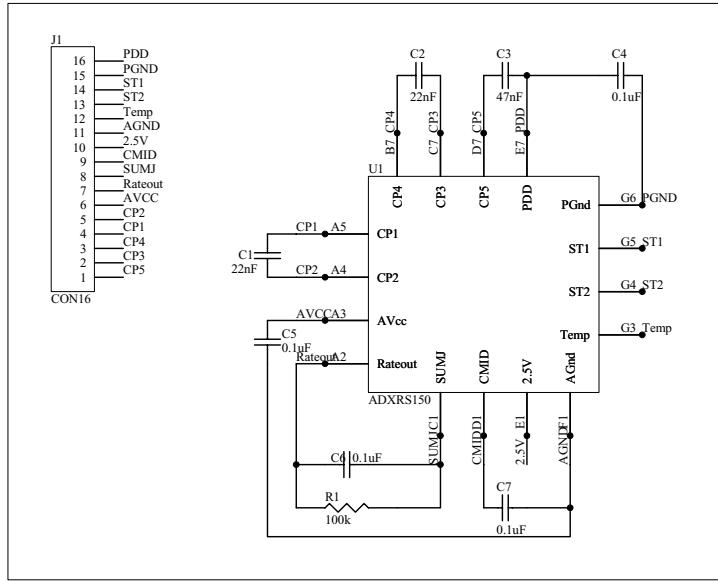


Figure D.4 Pin mappings for the commercially available ADXRS150

D.1.2 Tactile Board, Rev. 5

The schematic for the Tactile board (Rev. 5) is shown in Figure D.5. The board layouts with part placement information are shown in Figure D.6 (top) and Figure D.7 (bottom). Table D.1 lists the header pin to insole sensor mapping used for the GaitShoe insoles (all odd-numbered pins are connected to ground). Pins 1-4 can be used to connect to a ground plane, though the connection through the electric field sensor header can also be used.

TABLE D.1 Insole sensor mapping

Header Pins	Connection	Left Insole	Right Insole
1, 2, 3, 4	Ground		
5, 6	FSR	Lateral heel	Medial heel
7, 8	FSR	Fifth (lateral) metatarsal	First (medial) metatarsal
9, 10	PVDF	Calcaneous (heel)	Great toe
11, 12, 13, 14	Bend Sensor	Insole	Insole
15, 16	PVDF	Great toe	Calcaneous (heel)
17, 18	FSR1	Medial heel	Lateral heel
19, 20	FSR	First (medial) metatarsal	Fifth (lateral) metatarsal
21, 22, 23, 24	Bend Sensor	Ankle	Ankle

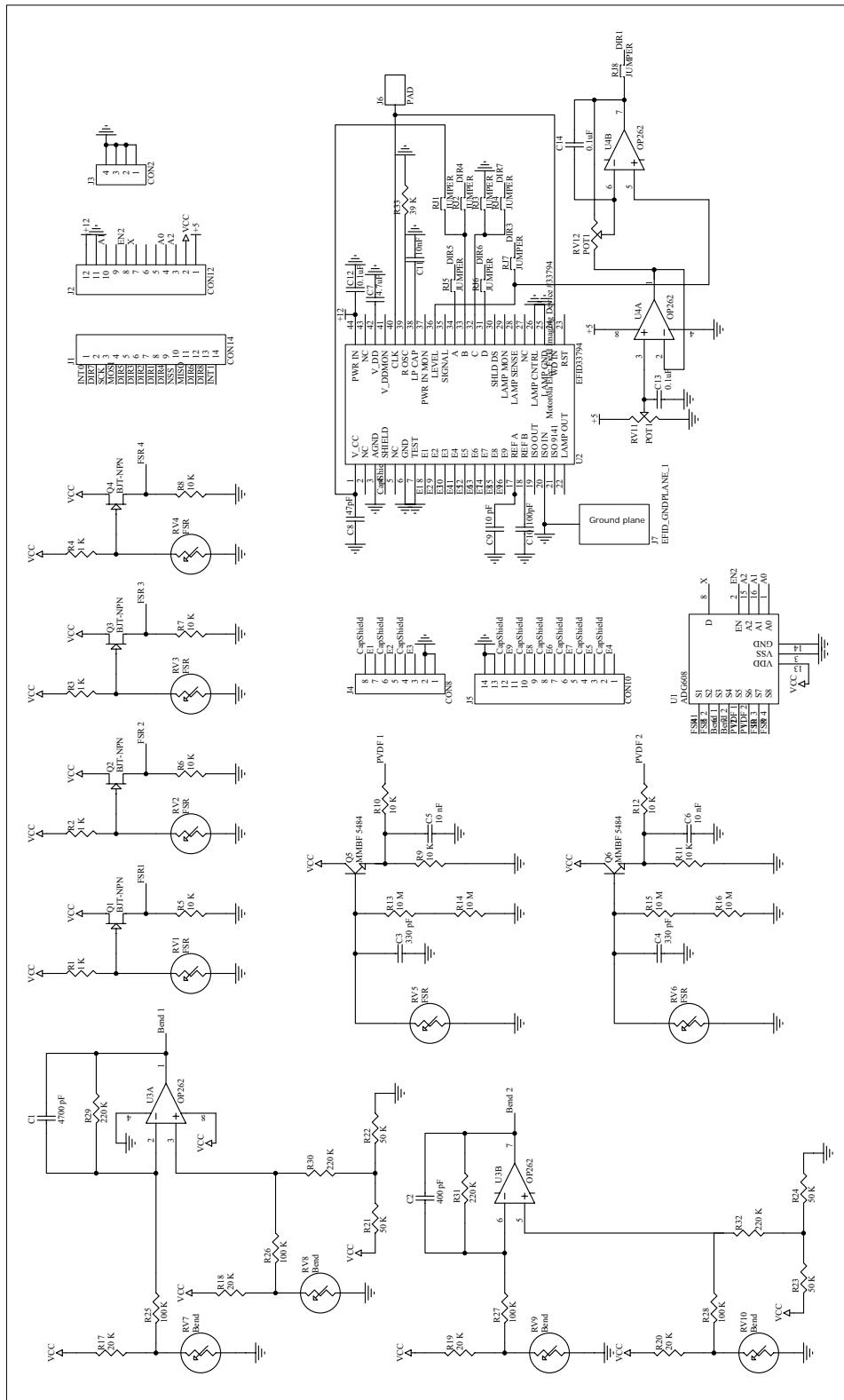


Figure D.5 Schematic of the Tactile board

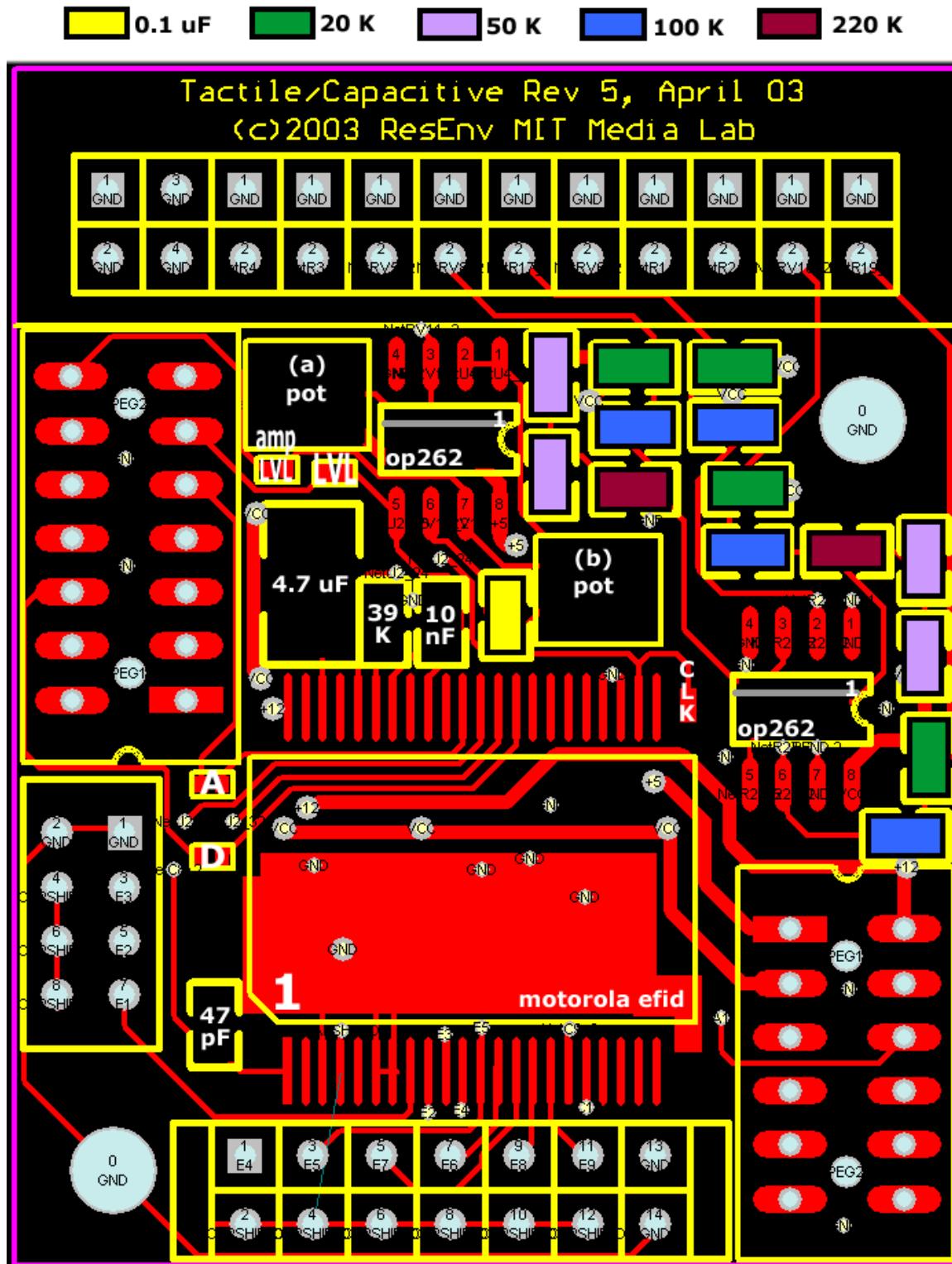


Figure D.6 Layout of the top side of the Tactile board

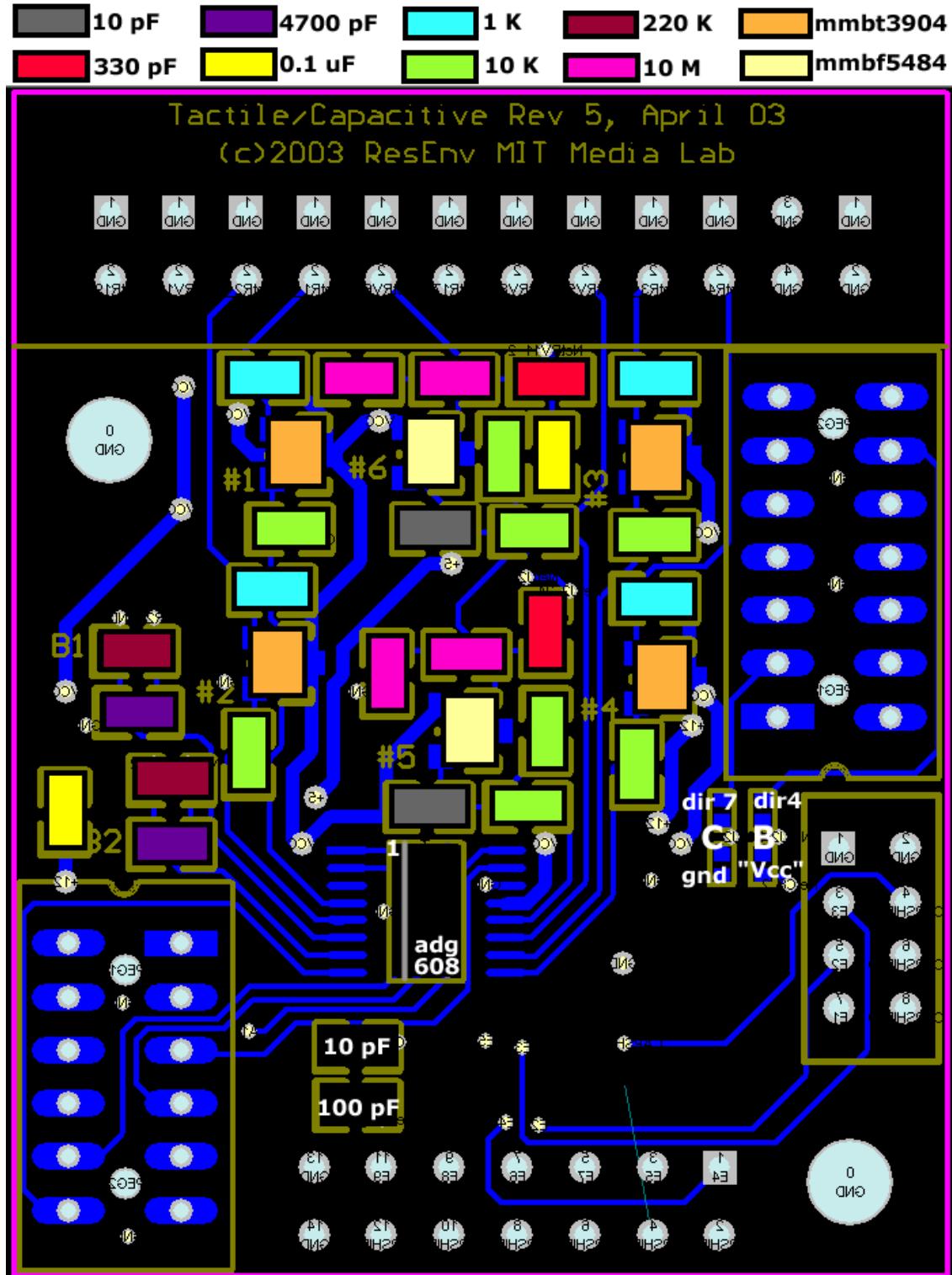


Figure D.7 Layout of the bottom side of the Tactile board

D.1.3 Main Board, Rev. 5

The schematic for the Main board (Rev. 5) is shown in Figure D.8. The board layouts with part placement information are shown in Figure D.9 (top) and Figure D.10 (bottom).

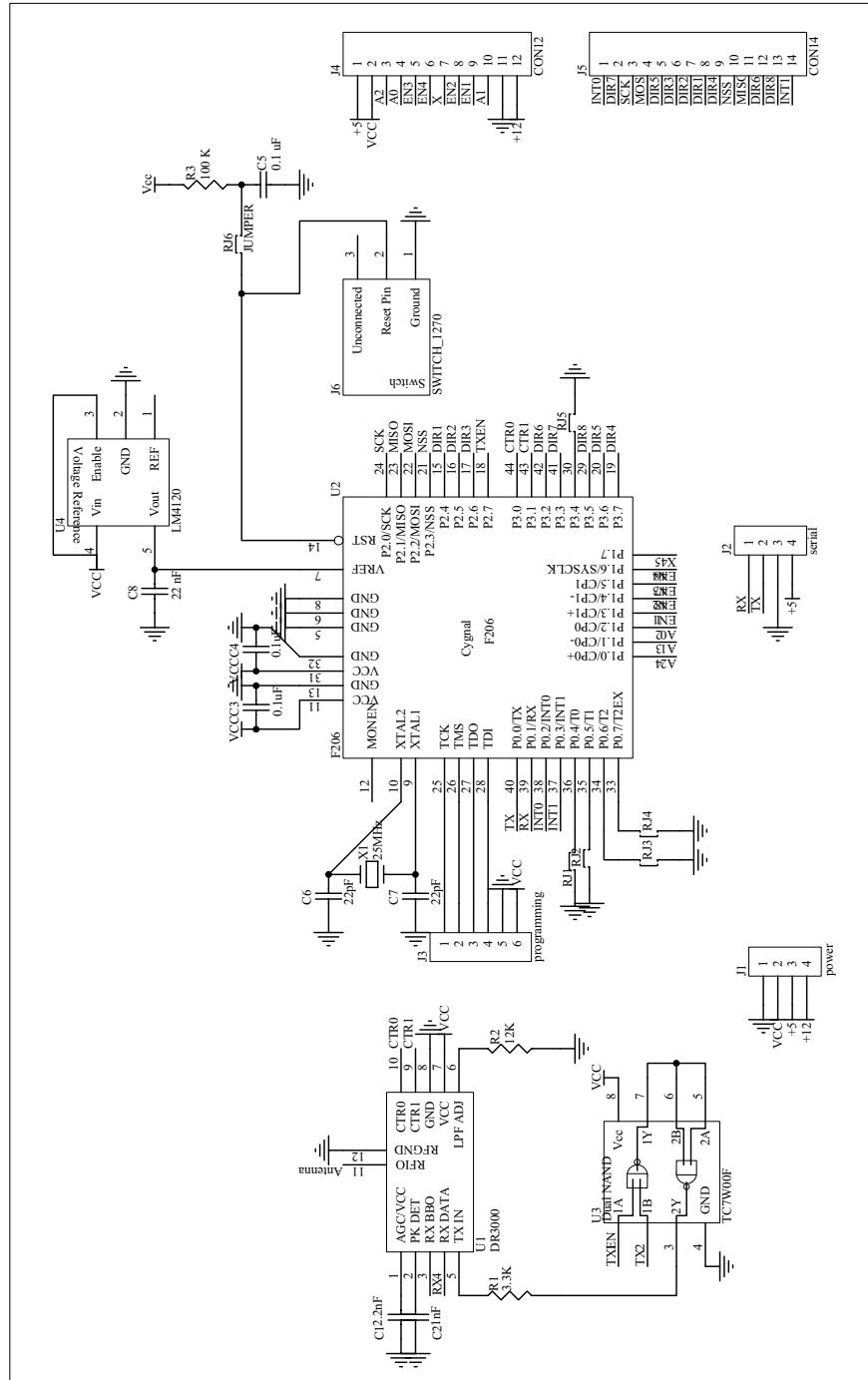


Figure D.8 Schematic of the Main board

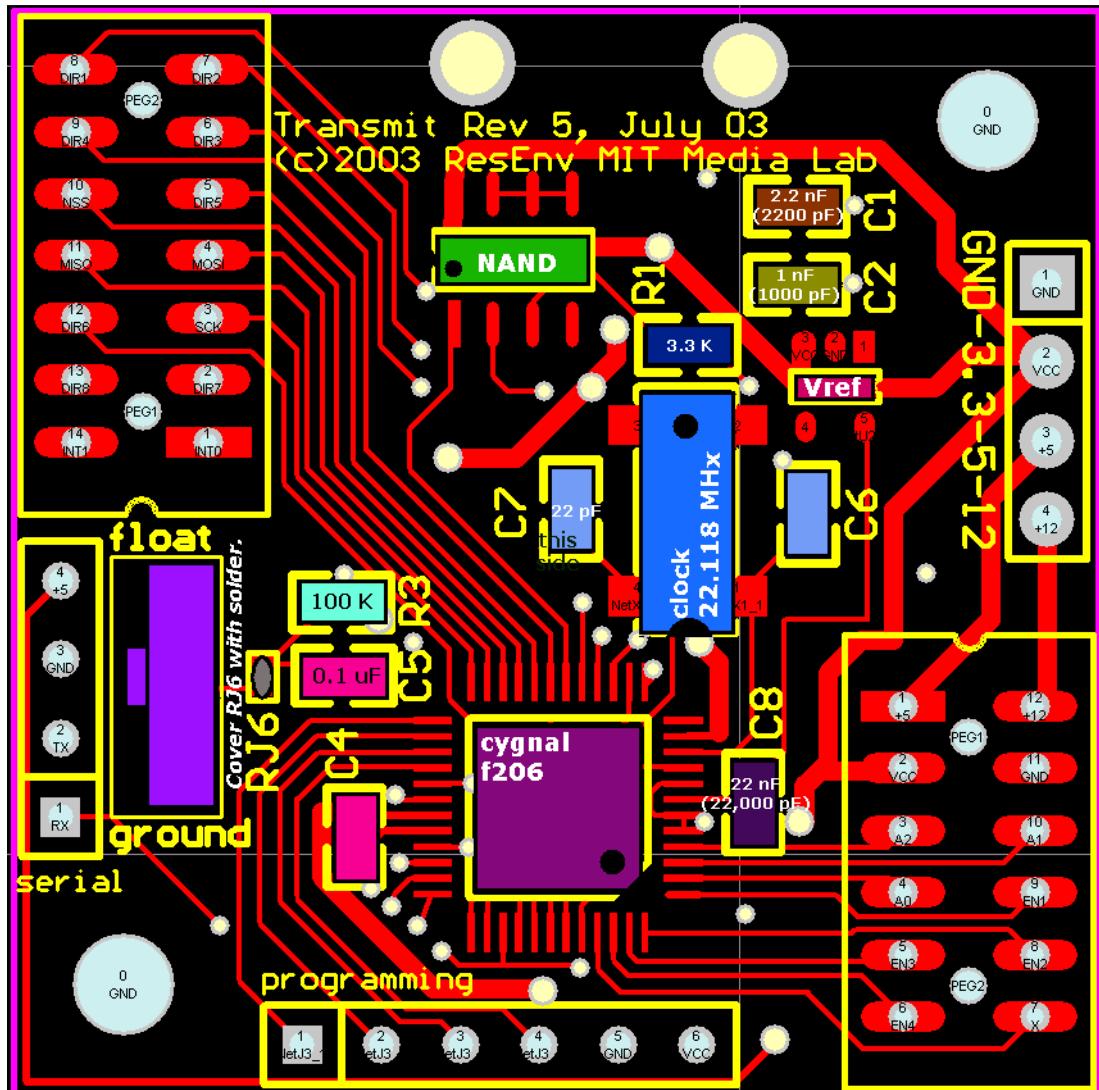


Figure D.9 Layout of the top side of the Main board

The six holes along the bottom edge of the Main board are for interfacing with the JTAG programmer, for uploading code to the microcontroller. The four holes along the left edge (looking at the top of the board) can be used to bypass the wireless transceiver and send the data directly to the computer. Both of these interface with the Programming board, described in Section D.1.5 below.

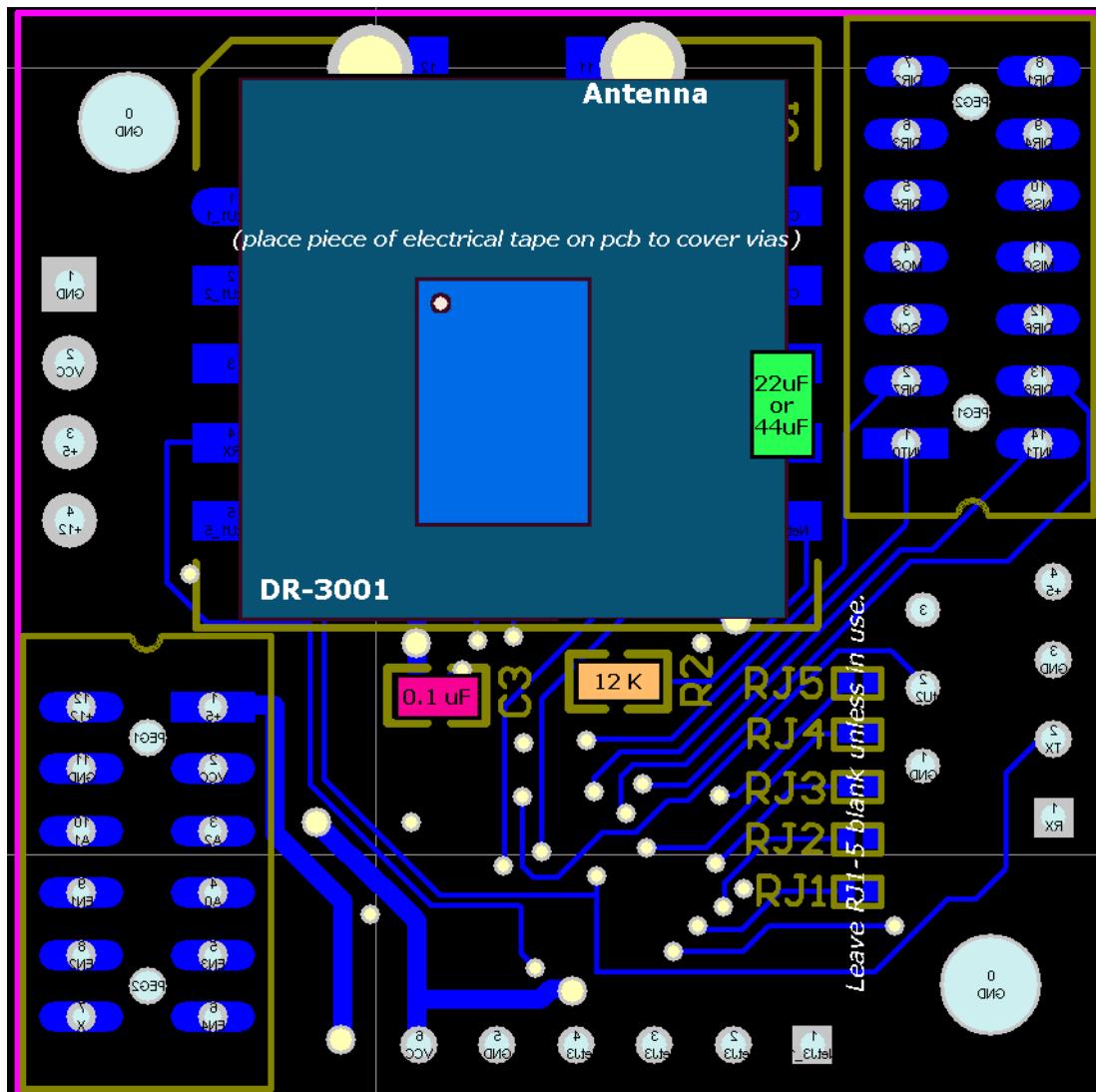


Figure D.10 Layout of the bottom side of the Main board

D.1.4 Power Board (Rev. 2)

The schematic for the Power board (Rev. 2) is shown in Figure D.11, and the board layouts with part placement information are shown in Figure D.12 (top) and Figure D.13 (bottom).

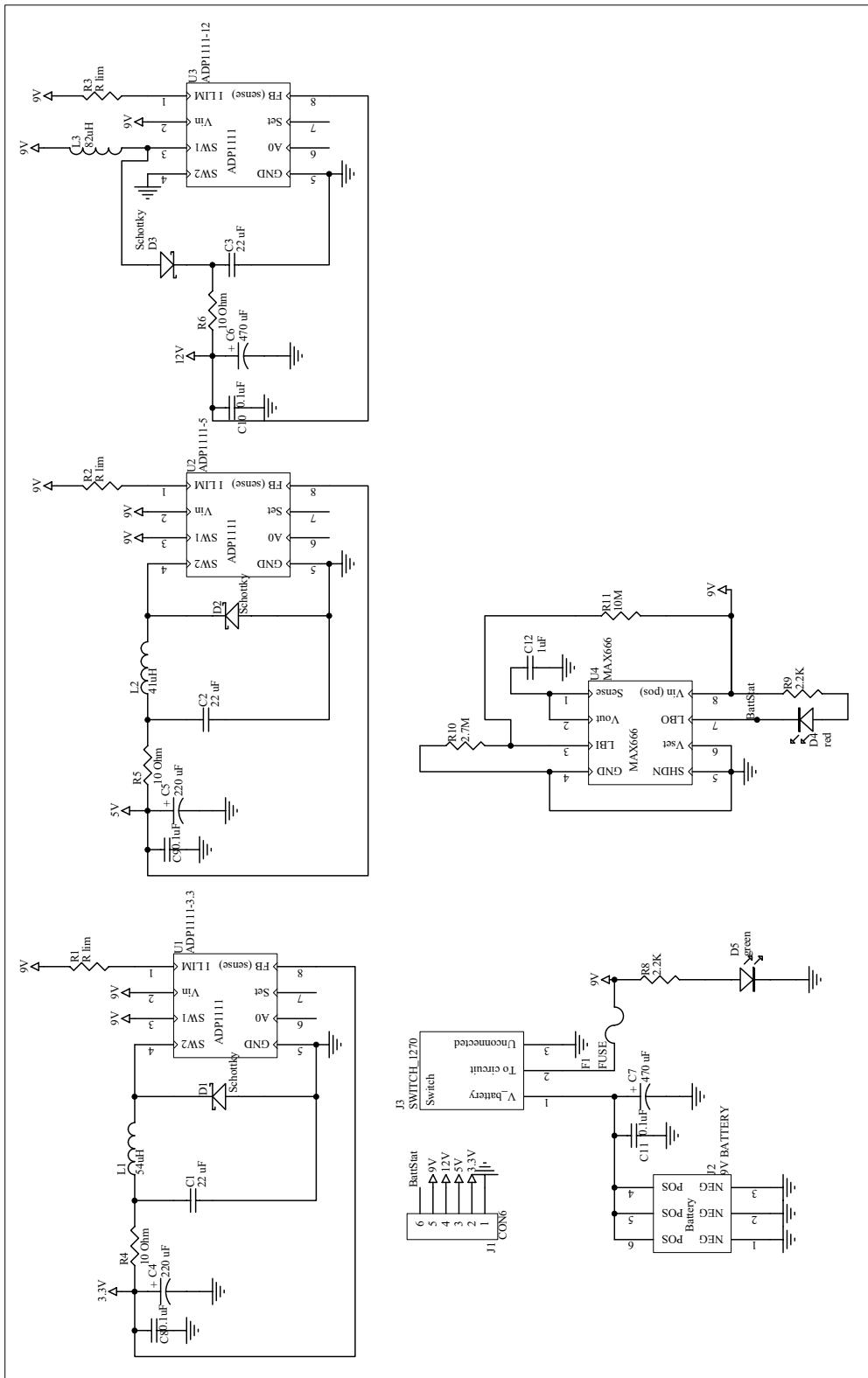


Figure D.11 Schematic of the Power board

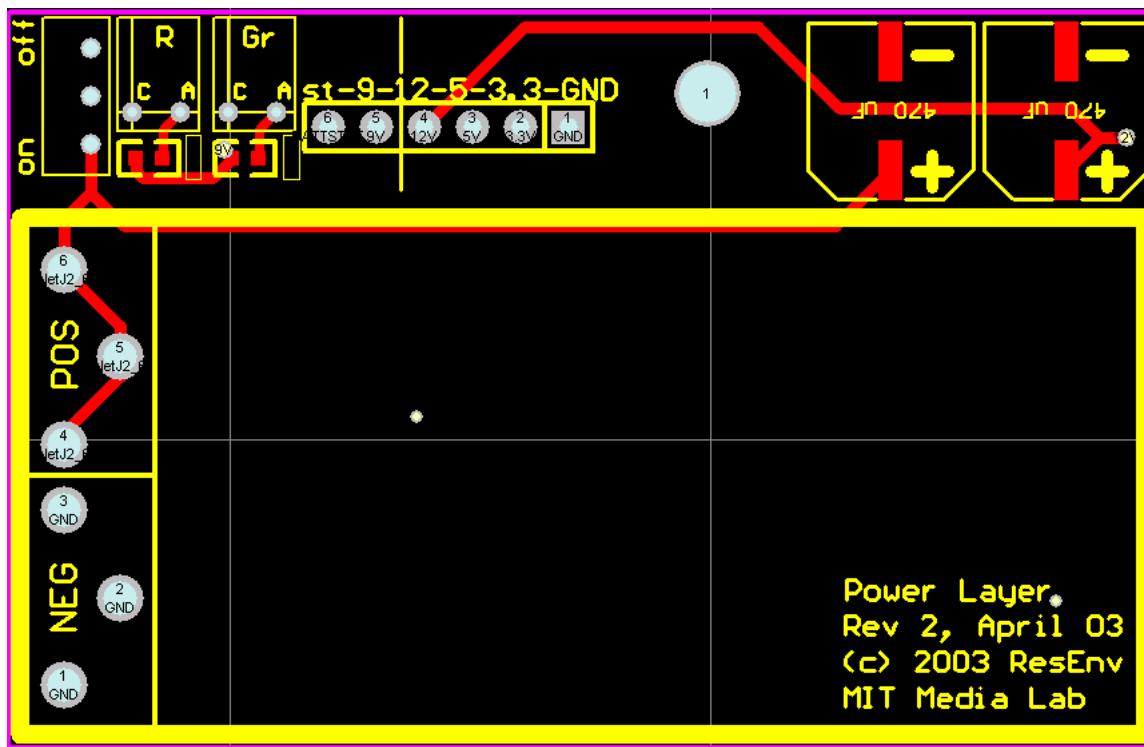


Figure D.12 Layout of the top side of the Power board

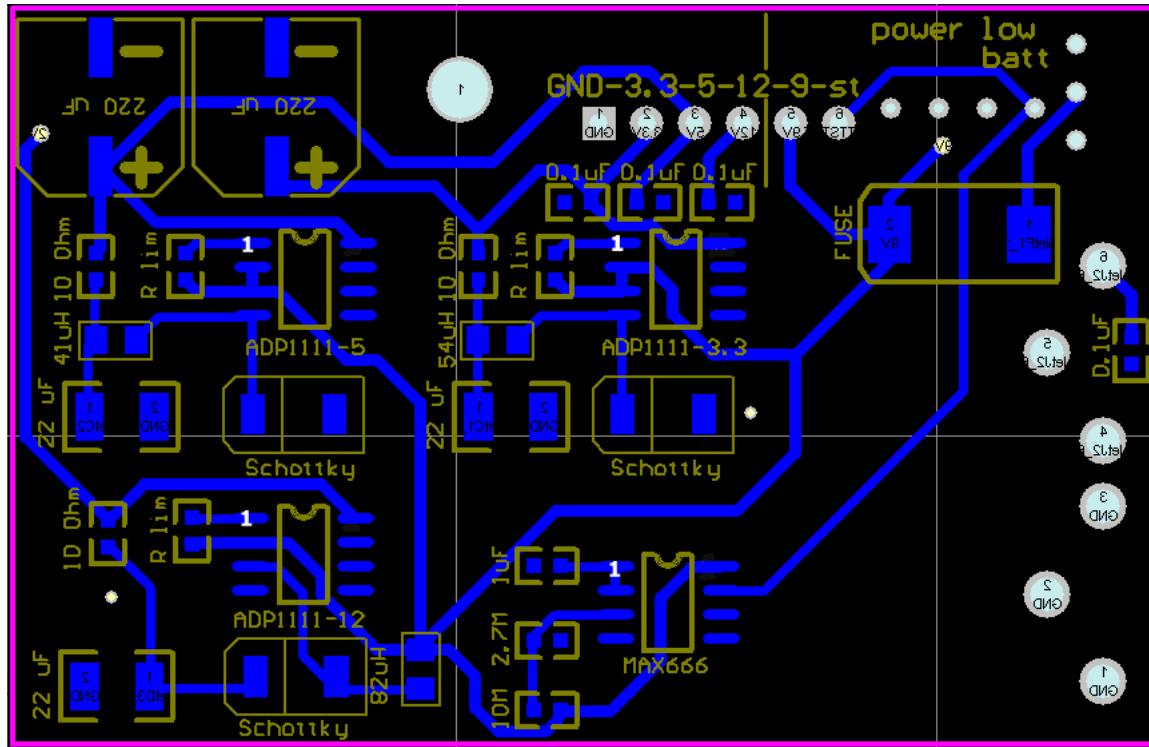


Figure D.13 Layout of the bottom side of the Power board

D.1.5 Programming Board (Rev. 2)

The Programming board was designed by graduate student Ari Benbasat. The schematic for the Programming board (Rev. 2) is shown in Figure D.14. The board layouts with part placement information are shown in Figure D.15, showing traces on both the top and bottom are shown).

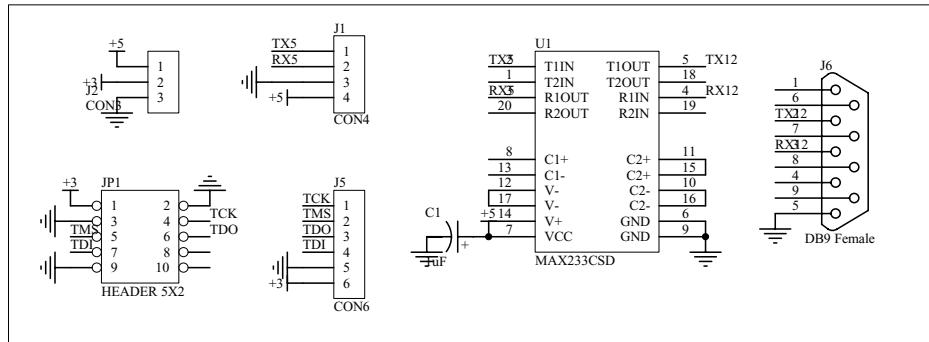


Figure D.14 Schematic of the Programming board

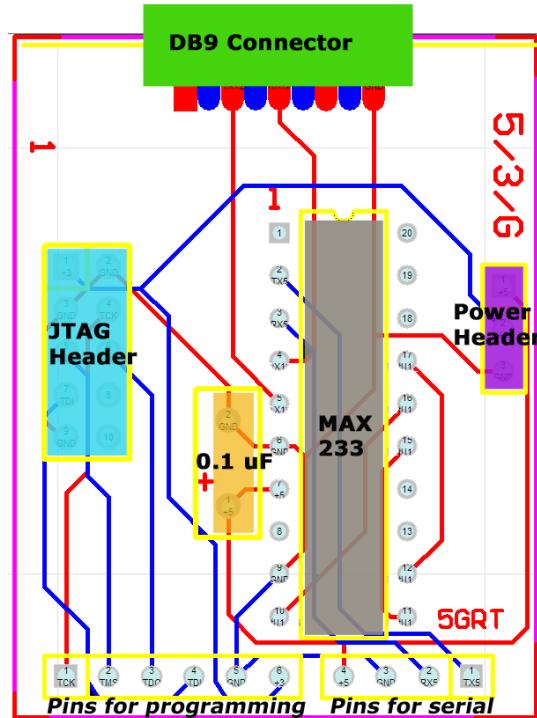


Figure D.15 Layout for the Programming board

D.2 Component Information

Information about the components used to build the GaitShoe is split across two pages, in Figure D.16 and Figure D.17. The part type, description, and a vendor (the websites for the vendors are listed in Figure D.2) and the part number for the vendor are listed, as well as the footprint and designator corresponding to the schematics of each board, and any relevant comments.

Part Type	Description	Part Number	Vendor	Footprint	Pad Designator	Board	Comments
ADG608	8 Channel MUX			TSSOP16	U2	IMU5	
916 MHz	Antenna	ANT-916-CW-RCS	Digkey	---	--	tactile4	
9V	Battery	P145	Digkey	SMBATTARY-FULL	J2	power2	
Band Resistor		FL1-01	Digkey	302-33-0PN	RV1, RV2, RV3, RV4	IMU5	
B1J1EN		ED161804D1CT	Digkey	302-33-48	I1, I2, O3, O4	tactile4	
C3PA1L226		PC171203A-1	Digkey	403	C1, C2, C3, C6, C9, C13, C14	IMU5	
Capacitor		PC17162-CT	Digkey	403	C15, C16, C17, C18, C19, C20,	IMU5	
1 nF (1000 pF)	Capacitor	PCC1028VCCT	Digkey	603	C21, C22, C23, C24, C25	IMU5	
1 uF	Capacitor	PC2224-CT	Digkey	603	C8, C9, C10, C11	power2	
2.2 nF (22,000 pF)	Capacitor	PC17174CT	Digkey	603	C3, C4, C5	IMU5	
22 nF (22,000 pF)	Capacitor	PC17194CT	Digkey	603	C12	power2	
22 pF	Capacitor	PCC220ACVCT	Digkey	1210	C1, C12	IMU5	
22 uF	Capacitor	PC22307CT	Digkey	603	C6, C7	power2	
220 uF	Capacitor	PCE3400CT	Digkey	603	C1, C2, C3	IMU5	
47 nF (47,000 pF)	Capacitor	PC17156CT	Digkey	603	C4, C5	power2	
47 uF	Capacitor	PC22231CT	Digkey	603	C4, C5, C7, C8, C10	IMU5-extra	
4.70 uF	Capacitor	PC3402CT	Digkey	603	--	power2	
4.70 uF	Capacitor	PC17140CT	Digkey	603	C6, C7	IMU5	
4.70 uF	Capacitor	SE2707C-T	Digkey	603	C1, C2	tactile4	
22.118 MHz	Clock	A26808	Digkey	603	X7L-SMD	IMU5	
24 pin	Connector	593K	Digkey	603	--	power2	
94+ Terminal	Connector	CON4	Digkey	603	--	power2	
94+ Terminal	Connector	CON4	Digkey	603	--	power2	
Connector CON4	Connector CON4	CON4PSMA0095C08.5	Digkey	5976/35	5976/60/G	IMU5	Used this one with current transit board. Only need one between the bands, not one per band!
CON6	HEADER 3	WM2737	Digkey	SIP68IG	J1	power2	
CON6	HEADER 3	ED7534-ND	Digkey	SIP68IG	J1	power2	
CON12	20 Pin socket	WM18078	Digkey	87332-12	J1	power2	
CON14	Connector-female	WM18019	Digkey	87332-14	J4	power2	
CON12	Connector-Male	WM18077	Digkey	87332-12	J3	power2	
CON14	Connector-Male	WM18078	Digkey	87332-14	J4	power2	
F206	Development Kit	334-1040	Digkey	---	J5	power2	
F206	Electric Field Imaging Device	MCG3794DH	ISDPA44	---	power2	IMU5	Programmable extra. Buy one dev kit to get the serial adapter for programming.
FSR	Force Sensitive Resistor	409 or 402	Intelink Technology	SAB2	U2	power2	
Fuse		MS-SK400CT	Digkey	RUE	I1	IMU5	Needed an interface built to IMU5 (see Figure D.4).
AN205150	Gyro (Azimuth)	ENCD24-M	Digkey	ENCDOM-SMT	J5	IMU5	You need two, one "A" type, one "B" type. (not been used yet)
ENC23M	Gyro (Magnet, surface mount)	ENCD24-J	Digkey	ENCO2L-PINS	J1, J2	IMU5	You need two, one "A" type, one "B" type.
ENC033	Gyro (Magnet, through pins)	AKR24A	Digkey	---	--	tactile4	Used to connect to the IMU5 sensors.
24 pin	Header						

Figure D.16 Component information, part one.

Part Type	Description	Part Number	Vendor	Footprint	Pad Designator	Board
18 uH	Inductor	TK53625CT	Digkey	1008	L2	power2
27 uH	Inductor	TK53628CT	Digkey	1008	L1	power2
82 uH	Inductor	TK53634CT	Digkey	1008	L3	power2
Programmable Resistor	... Interface Module	... MBF1666	... ON Semiconductor	SP1a	... C6 D6	main5 main5 tactile4
SMA Header	... MBF1666	MBF5484L71	ON Semiconductor	SO75TAC72 (SOT-23)	... D5	power2
JFET	... LED	67-1240	Digkey	... P075TAC72 (SOT-23)	... D4	main5
Green LED	... Red LED	67-1238	Digkey	P075TAC72 (SOT-23)	... D3	power2
Red LED	... Inert Accelometer (2 axis, 2G)	A0X1202IE	Albany Devices	... SC88-P-1.27	... U3	main5
ADAC20E	... Tilt Sensor	ADAC20E	Albany Devices	... SC88-P-1.27	... U2	power2
MC34063	... Nuts	91B95A05	McMaster Carr	... SC88-P-1.27	... U1	main5
Acorn Nut	... Op-amp	OP262	Analog Devices	SC-8	U6, U7, U8	Used for securing Power board to the attachment. Used to hold to attachment.
1/4" thread	... Op-amp	OP262GS	Analog Devices	... SC-8	U1, U6, U7, U8	Used for securing Power board to the attachment. Used to hold to attachment.
PVC Film	Plastic Film	B2975K17	McMaster Carr	... JWMSQ-SMDPOT	R6, R5, R6, R7	tactile4
Pt100	Polyimide Diffused Potentiometer	B2975K12	McMaster Carr	... JWMSQ-SMDPOT	R1, R2, R3, R5, R7	main5 main5 tactile4
100K Pot	... Potentiometer	PEC104CT	Digkey	... JWMSQ-SMDPOT	R1, R2, R3, R5, R7	main5 main5 tactile4
200K Pot	... Potentiometer	PL204CT	Digkey	... JWMSQ-SMDPOT	R1, R2, R3, R5, R7	main5 main5 tactile4
0 K	... Resistor	PL100CT	Digkey	... JWMSQ-SMDPOT	R1, R2, R3, R5, R7	main5 main5 tactile4
1 K	... Resistor	PL01KCT	Digkey	... JWMSQ-SMDPOT	R1, R2, R3, R5, R7	main5 main5 tactile4
10 K	... Resistor	PI0.01KCT	Digkey	... JWMSQ-SMDPOT	R1, R2, R3, R5, R7	main5 main5 tactile4
10 M	Resistor	311-10MGCT	Digkey	603	R2, R4, R6, R8, R11, R14	tactile4
10 M	Resistor	311-10MGCT	Digkey	603	R5, R7, R10, R12, R13	power2
10 Ohm	Resistor	PI0.01KCT	Digkey	603	R1, R5, R6, R8, R10, R12, R13	power2
100 K	Resistor	PI0.01KCT	Digkey	603	R20, R22, R28, R30	tactile4
				R3	... R3	main5
12 K	Resistor	311-12OKHCT	Digkey	603	R2, R4, R6, R8, R10, R12, R13	main5
12.4 K	Resistor	PI1.12OKHCT	Digkey	603	R5, R7, R9	main5
13 K	Resistor	311-13OKHCT	Digkey	603	R5, R7, R9	main5
15 M	Resistor	PI5MGCT	Digkey	... 603	R5, R7, R9	main5
2.2 K	Resistor	311-2.2OKHCT	Digkey	603	R5, R7, R9	main5
2.7 M	Resistor	P2.7MGCT	Digkey	603	R10, R12, R27, R29	tactile4
20.3 K	Resistor	PI2.03OKHCT	Digkey	603	R13, R26, R31, R34	main5
3.3 K	Resistor	P3.3MGCT	Digkey	603	R13, R26, R31, R34	main5
5 M	Resistor	P5.1MKCT	Digkey	... 603	R14, R25, R32, R33	tactile4
50 K	Ferrite Bead	P810CT	Digkey	1210	R18, R19	main5
500 K	Surface Diode	SD100CT	McMaster Carr	... SD100CT	D1, D2, D3	power2
5.1 K	Surface Diode	SD1722A18	McMaster Carr	... SD1722A18	... D1, D2, D3	power2
5.18" length, 4.40s	Screws	91772A06	McMaster Carr	... 91772A06	... D1, D2, D3	power2
1/4" length, 4.40s	Screws	209F	McMaster Carr	... 209F	... D1, D2, D3	power2
DB9 Female	Serial Connector	MAX233.CPP	McMaster Carr	... 8459A209	... D1, D2, D3	power2
RS-232	Serial Driver/Receiver	AS5124	McMaster Carr	... 8459A209	... D1, D2, D3	power2
1.27" length	Spacers	AS5124	McMaster Carr	... 8459A209	... D1, D2, D3	power2
24 pin	Strain Relief	EG1847	Digkey	... SWITCH1270	... J3	main5
SWITCH-1270	Voltage Reference	LM4120	Digkey	... SO723-5	U4	power2
Voltage Regulator	Voltage Reference	LM4204AM5-3.OCT	Albany Devices	... SO-8	U3	main5
Wireless Transceiver	Voltage Regulator	ADP1111AR-12	Albany Devices	... SO-8	U4	power2
DR3000-1	Voltage Regulator	ADP1111AR-3	Albany Devices	... SO-8	U4	main5
	Wireless Transceiver	MAX4665CA	RF Monolithics	DR3000-1	U1	(not printed in Rafts were created by the author?)

Figure D.17 Component information, part two

TABLE D.2 Vendor information

Company	Website Address
Analog Devices	www.analog.com
Android World	www.androidworld.com/prod47.htm
Digikey	www.digikey.com
Interlink Electronics	www.interlinkelec.com
McMaster Carr	www.mcmaster.com
ON Semiconductor	www.onsemi.com
RF Monolithics	www.rfm.com
Rochester Electronics	www.rocelec.com
The Images Co.	www.imagesco.com

D.3 Other Hardware Information

D.3.1 Insole Sensor Connections

The soldered connections to the sensors in the insole must be able to withstand the forces (normal and shear) encountered under the foot during gait. One method that has worked well was to cover the well-soldered connections with hot glue, let it cool slightly, press it flat, and then trim off the excess. This is demonstrated on a pair of bend sensors.

First the bend sensors are soldered to the wires, as shown in Figure D.18.



Figure D.18 Bend sensors soldered to wire

Next, a "gob" of hot glue is applied to the solder connections, as shown in Figure D.19.

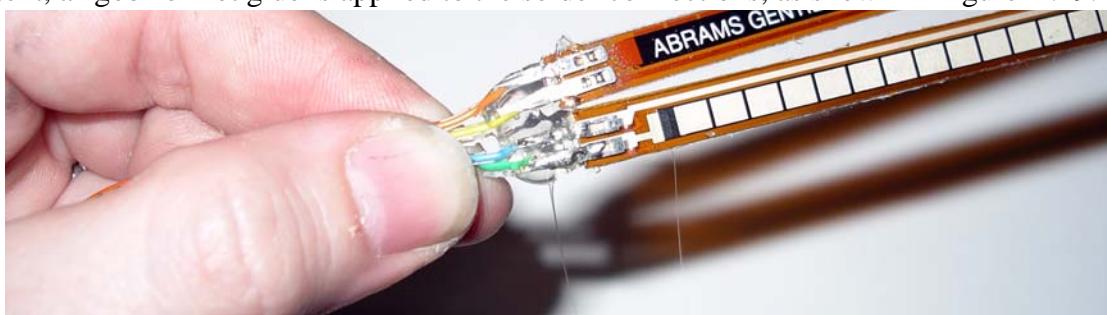


Figure D.19 Bend sensors with a "gob" of hot glue

After the hot glue cools slightly, it is pressed between fingertips to flatten it into a thin layer; the result is shown in Figure D.20.

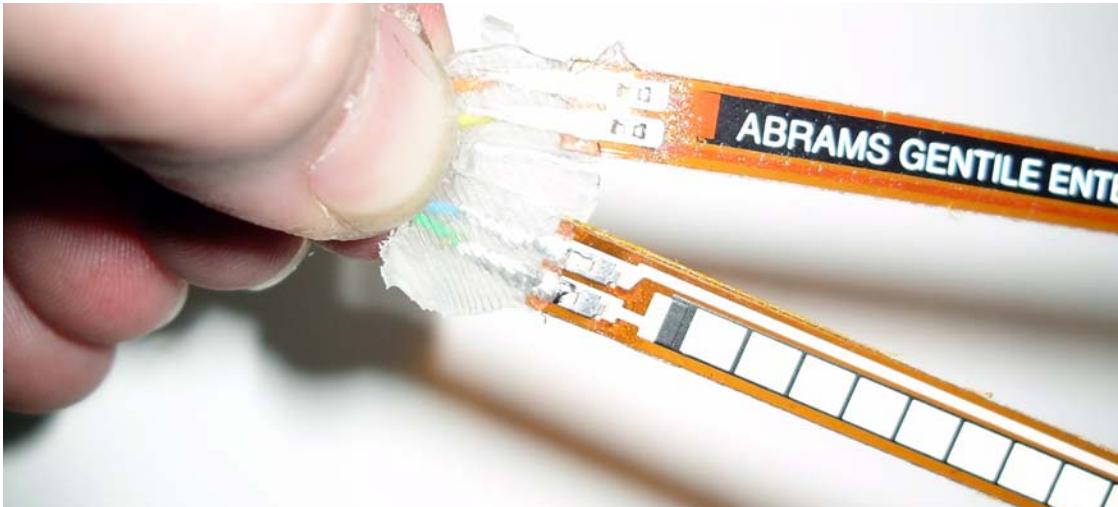


Figure D.20 Hot glue flattened by fingers

Finally, the hot glue is trimmed, as shown in Figure D.21.



Figure D.21 Trimmed hot glue on bend sensor solder connection

D.3.2 GaitShoe Attachment

The GaitShoe attachment was made out of 0.125" polyethylene terephthalate glycol (PTG), which is a thermoformable and machinable material. The process of making the attachment is detailed in the photos below.

First, a pattern, such as shown (to scale) in Figure D.22, can be used to trace the pattern of the attachment on to the PTG, as shown in Figure D.23. The solid lines in Figure D.22 indicate the base pattern and the dashed lines indicate extra material to add for a right or left antenna attachment (the pattern does not need to be followed exactly).

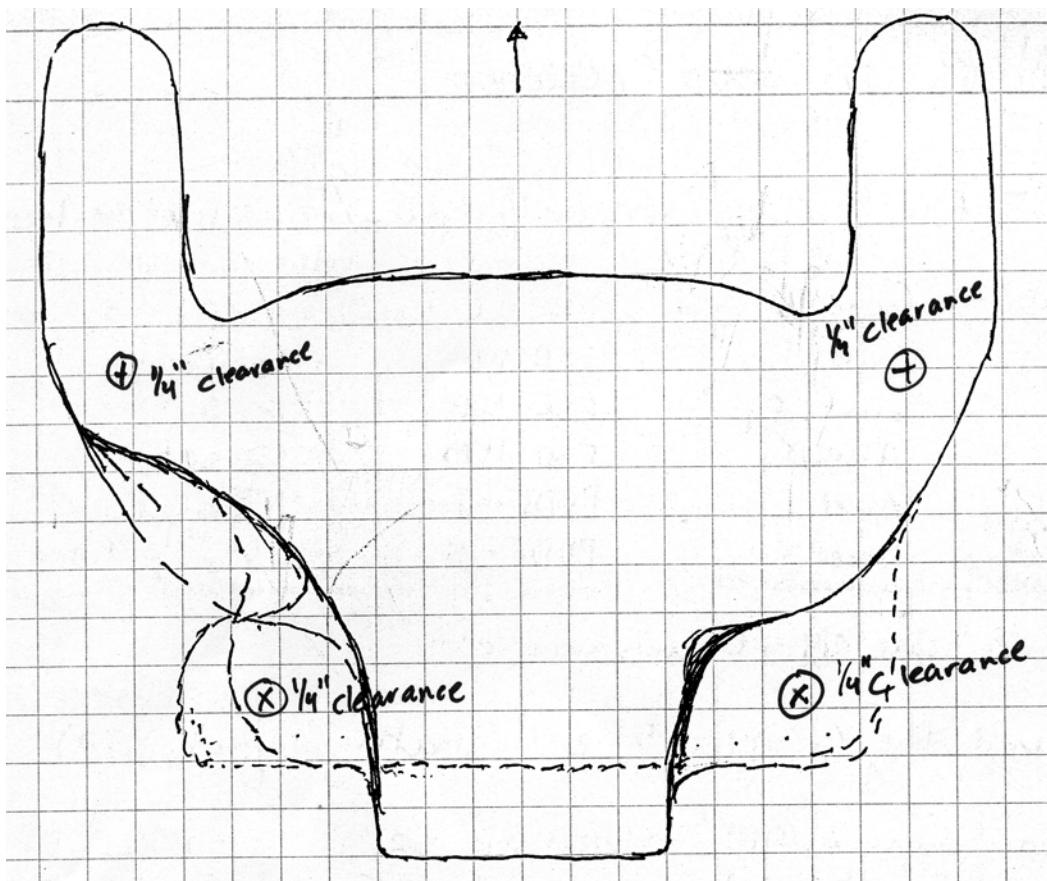


Figure D.22 Sketch of the pattern for both the left and right GaitShoe attachments

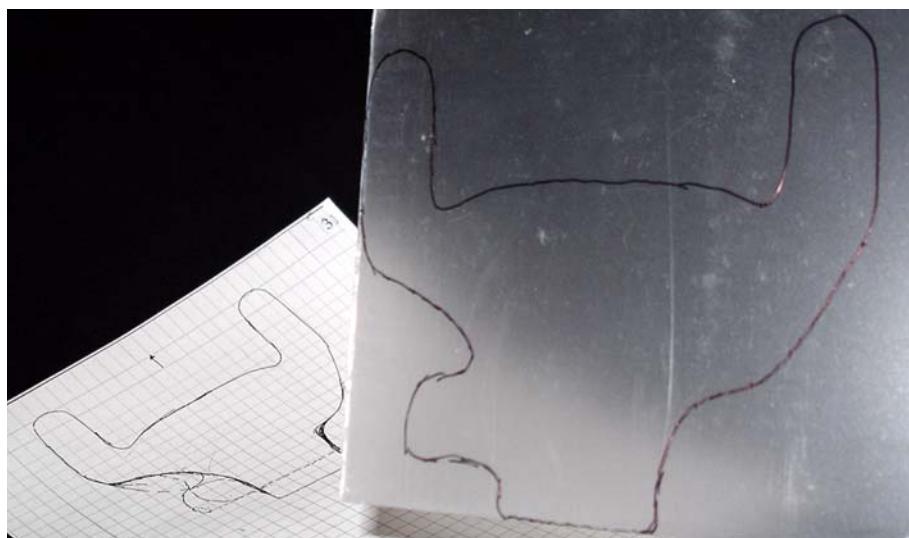


Figure D.23 Pattern traced onto PTG for forming the GaitShoe Attachment

Next, the easiest way to cut the PTG is to use heavy-duty shears, as shown in Figure D.24.

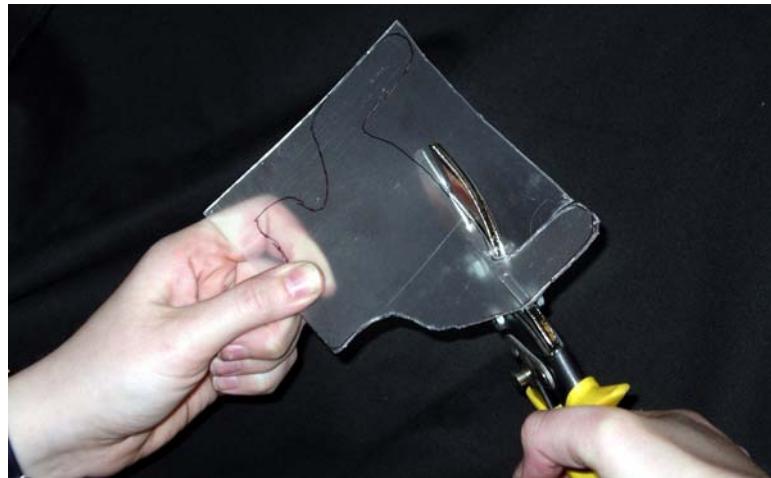


Figure D.24 PTG cut to shape with heavy duty shears

After the PTG is cut, the holes should be drilled¹, and then the PTG can be formed using a heat gun and a rounded block (or other available implement²), as shown in Figure D.25, to shape it to the desired geometry.



Figure D.25 Heat gun and wooden block used to shape the PTG

-
1. The final hole in the attachment connecting the battery enclosure to the attachment must be drilled after both pieces are shaped; the holes for the stack should be marked using the IMU of the stack which will be mounted on the attachment.
 2. For example, a the bottom of the attachment can be rolled around a small screwdriver to form the enclosure at the bottom of the attachment for the fishing line.

The battery enclosure can be formed similarly; a sample pattern is shown in Figure D.26.

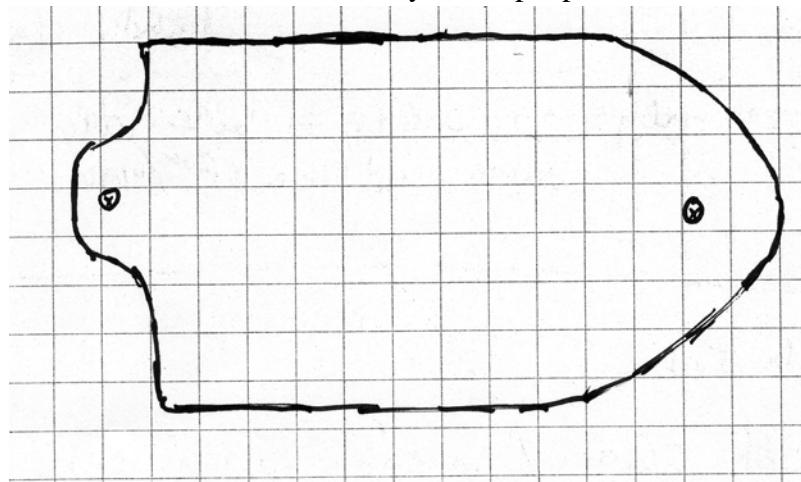


Figure D.26 Sketch of the pattern for the battery enclosure

A left GaitShoe attachment and battery enclosure are shown separately from the back, and together from the front in Figure D.27.

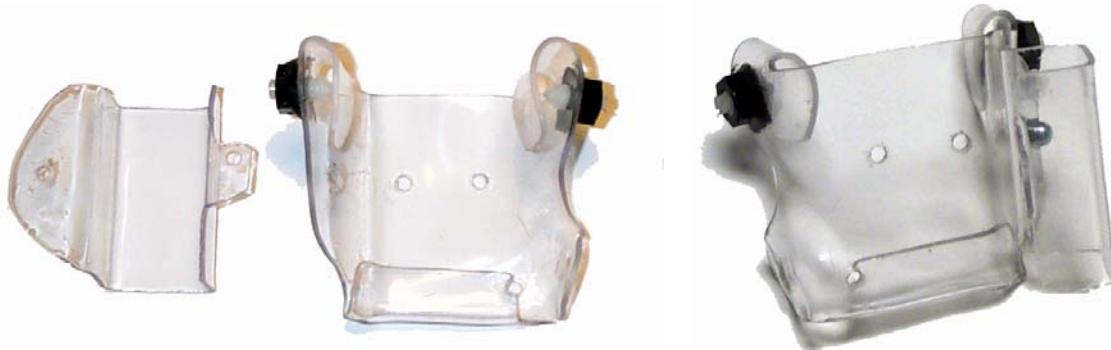


Figure D.27 GaitShoe attachment and battery enclosure; viewed separately from the back (left), and viewed together from the front (right)

Appendix E

ULTRASOUND SENSOR

This appendix describes in further detail the ultrasound sensor developed by Steven Dan Lovell; photographs of the ultrasound circuit boards are included in Section 3.2.1, and the working principle is discussed in Section 3.3.7. Mr. Lovell contributed to the writing of this appendix.

Ultrasound transducers can turn electrical energy into mechanical energy and mechanical energy into electrical energy. They have a range of frequencies over which they are resonant, where the ratio of energy input to energy dissipated in the device itself is low. The mechanical energy they produce and sense is in the form of pressure waves, thus the signal they can send propagates at the speed of sound, making them suitable for determining distances between a transmitter and a receiver by measuring the time between the transmission of the signal and the reception of the signal.

E.1 Circuit Boards

The ultrasound sensor was implemented on two circuit boards: a transmit board that attached to the stack on the right shoe, and a receiver board that attached to the stack on the left shoe. Attached directly to the transmit board was a single daughter board to hold the transmitter. The receive board had two daughter boards that each held a receiver, and were connected to the receive board via wires. These are shown in Figure E.1.

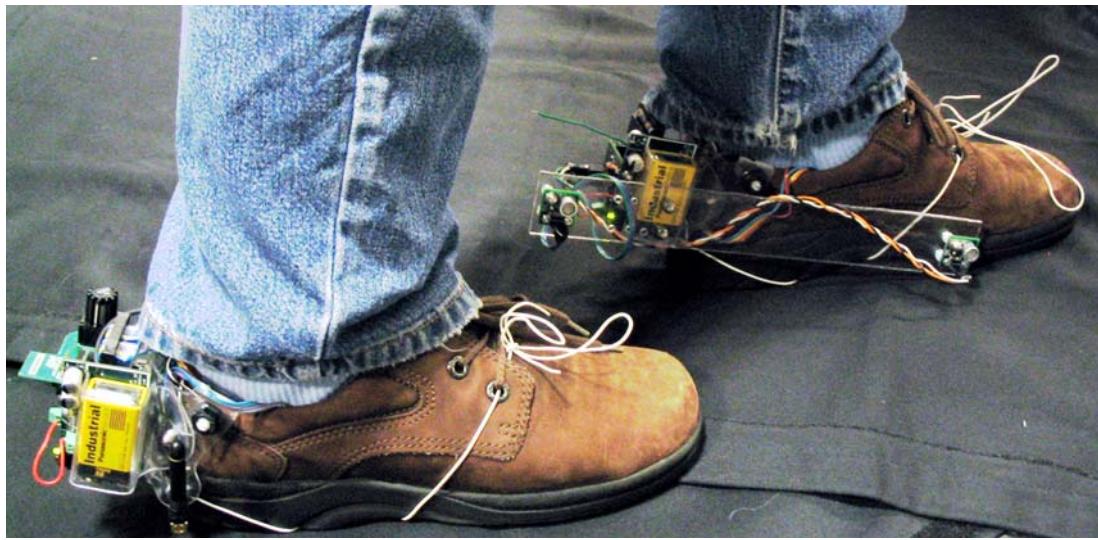


Figure E.1 Photo of ultrasound hardware mounted on the GaitShoe hardware

The schematic of the main transmit board is shown in Figure E.2, and the schematic of the transmit daughter board is shown in Figure E.3.

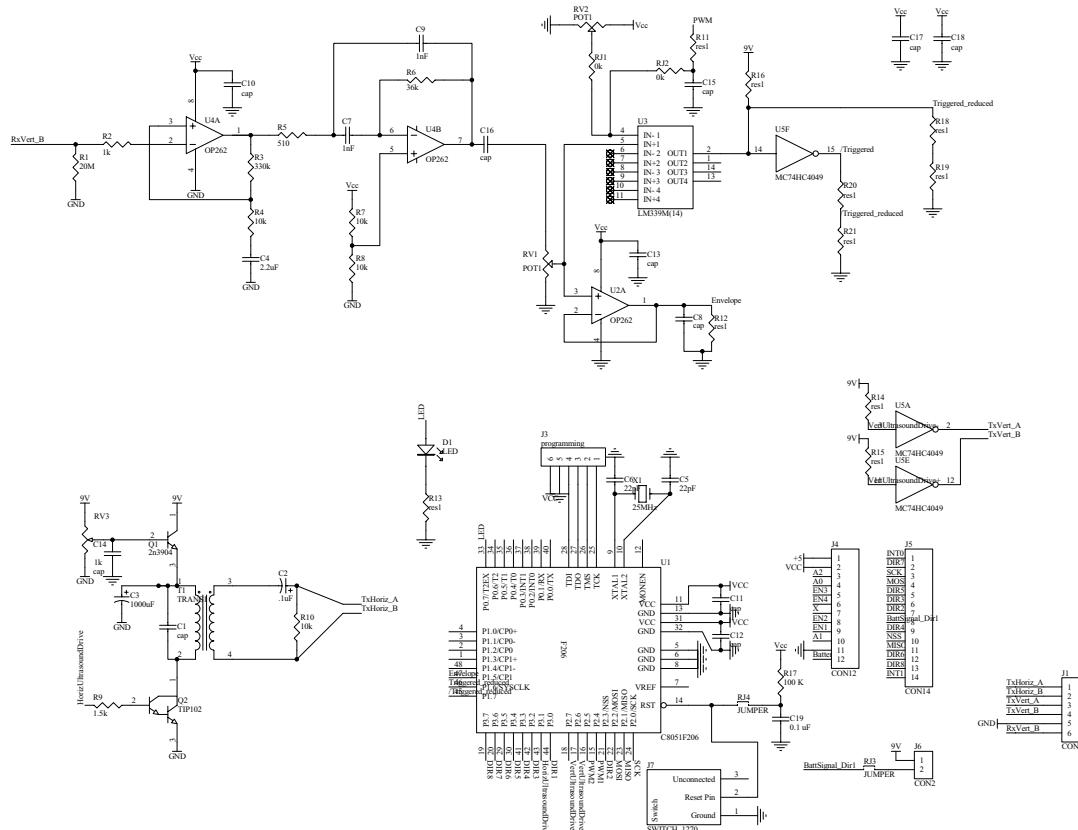


Figure E.2 Schematic of the ultrasound transmit board.

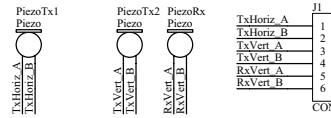


Figure E.3 Schematic of the ultrasound transmit daughter board

The schematic of the main receive board is shown in Figure E.4, and the schematic of the receive daughter board is shown in Figure E.5.

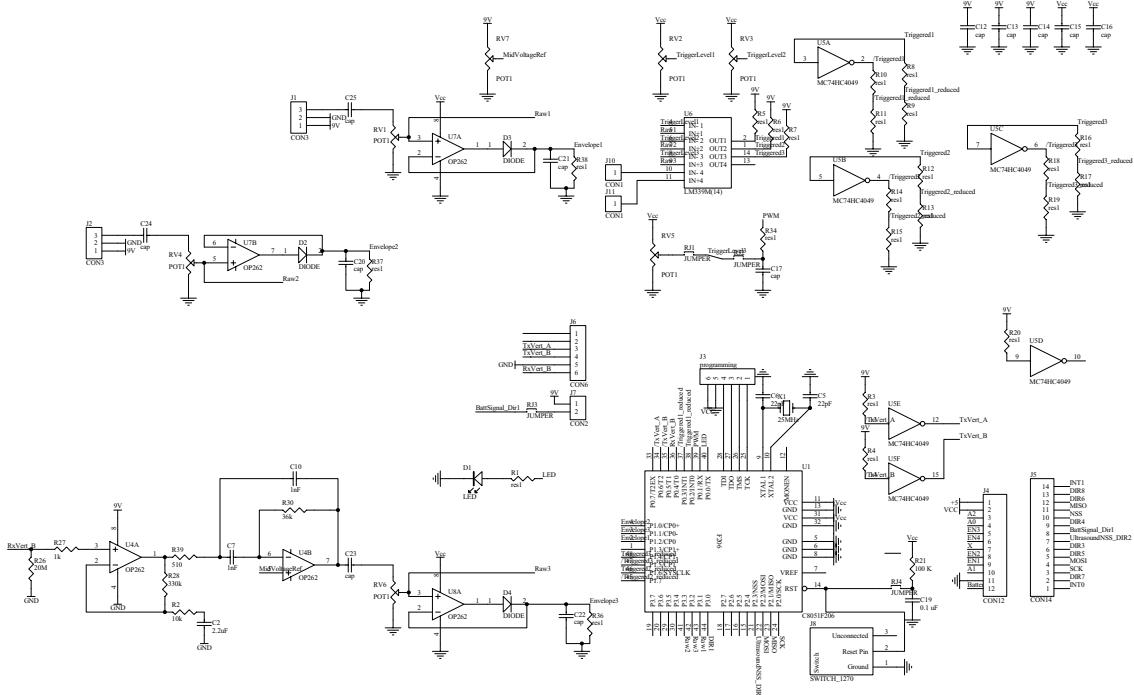


Figure E.4 Schematic of the ultrasound receive board.

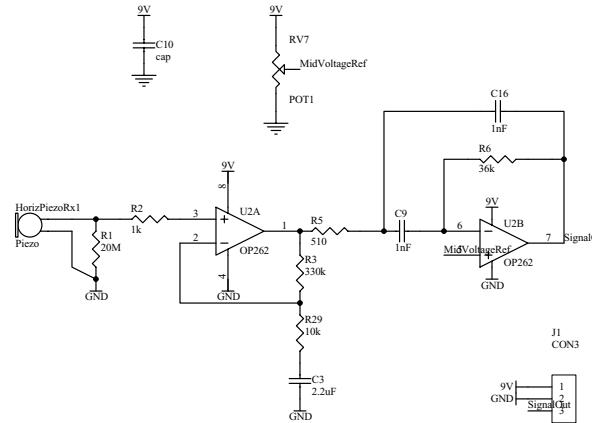


Figure E.5 Schematic of the ultrasound receive daughter board.

An omni-directional polyvinylidene fluoride (PVDF) ultrasound transducer manufactured by MSI, part 1005853-1, was used for the transmitter [120]. Two directional piezoelectric ceramic (PZT) ultrasound transducers (such as V-MA40A5R [121]) were used for the receivers.

The omni-directional transducer must be driven at high voltages (150 V-300 V) at 40 KHz to generate a signal that can be measured over a distance of a few feet. A step-up transformer, resonant at 40 KHz with the ultrasound transducer as a load, along with a push-pull amplifier that takes a 3.3 V input from the microcontroller and outputs 9 V, was used to drive the omni-directional transducer at 270 V.

The receivers were placed on signal conditioning daughter boards that could be placed away from the main ultrasound receiver board. The daughter boards had two stages of filtering: a medium gain single pole high-pass filter and a medium gain double pole band-pass filter that added DC bias. This signal was then fed back to the main receiver board where a comparator was used to determine when the signal crossed a threshold. The threshold is variable and set by a potentiometer. This threshold crossing is used to determine when the first arrival of the ultrasound signal occurs.

To determine the distance between the transmitter and the receiver, the time of flight (TOF) between the transmitter and receiver is measured. This TOF is multiplied by the speed of sound in air to determine the distance of separation between the transmitter and receiver. To determine the TOF, the transmission of the ultrasound signal and the start of timer on the receiver must occur simultaneously. For this, the wireless radio frequency (RF) transmission already employed for communication between the GaitShoe hardware and the basestation was used. The basestation issued the ultrasound transmission command to initiate the start of transmission and the start of the receiver timer; the length of time between the RF transmission from the basestation and the reception on the Main Stack board, and then the subsequent communication between the Main stack board and the ultrasound transmit and receiver boards was assumed to be the same for the transmit

and receiver boards. When the ultrasound transmit board received an ultrasound transmission command it drove a short pulse of 40Khz square wave to the ultrasound transducer. When the ultrasound receive board received an ultrasound transmission command it started a timer, counting the number of clock cycles. When the ultrasound receive board was signaled by the comparator that a threshold crossing occurred, the value of the timer (i.e. total number of clock cycles) was stored, and subsequently transmitted to the base station. This occurred twice, once for each receiver. The two distances can be post-processed to determine the angle at which the line between the transmitter and middle of the two receivers is oriented relative to the line perpendicular to the two receivers (see Figure 3.35).

E.2 Microcontroller Code

The microcontroller code written by Mr. Lovell for the transmit board is shown in Figure E.6 and for the receive board is shown in Figure E.7.

<pre> /* * ultrasound_tx.c - Cygnal F206 code for an ultrasound rangefinding transmit board * Written by S/I/03 for Mr. Lovell's MIT Lab Guidance Project * Variables named H_ are for physical measurements * Variables named V_ are for logical measurements * Upon receipt of the transmit command via SPI the transmit process is started * It consists of sending a timer that times the transmission frequency (using the timer * itself) and the duration of the ping (using a count of the number of timer interrupts) */ #include "206.h" // SPI Gen related declarations sbit R_SPI0_SCK; sbit LED_P07; int num_ints = 0; sbit P0_0; sbit V_Out2=P0_0; sbit clk = P1_4; #define V_OUTPUT_HIGH() V_Out1 = 1; V_Out2 = 0; #define V_OUTPUT_LOW() V_Out1 = 0; V_Out2 = 1; #define V_OUTPUT_1(V_Out1,V_Out2) V_Out1 = V_Out1; V_Out2 = ~V_Out2; #define V_OUTPUT_2() V_Out1 = 0; V_Out2 = 0; #define sysclk 2218000 // SPI related declarations char ULTRASOUND_START_CODE = 0x96; char SPI_In; char SPI_Out; void SPIint(void) interrupt 6 { SPI_In=SPIODAT; if(SPI_In==ULTRASOUND_START_CODE){ START_TIMER(0); H_Out = 1; PRT2CF = 0x01; SPIIF=0; } } void SPIinit(void) { SPIODAT = 0x00; //enable global interrupts SPIFCN = 0x01; //enable SPI in SLAVE mode SPIDCP = 0x07; //SPI Frame size = 8bits, this is the default setting SPIOSC = 0x07; //SPI oscillator rate = 1MHz, note that max is 20, maximum value is SYSCCLK/20 IEIE = 0x01; //Enable SPI interrupts PRT2NK = 0x01; //SPI pins SCK,MISO,MOSI,NSS available on P2 pins P2.0,P2.1,P2.2,P2.3 respectively PRT2CF = 0x02; //SCK,MOSI,NSS are input; MISO is output PRT2CF &= ~0xD; } void oscinit(void) { int delay; OSCXCN = 0x66; // Enable external crystal WDTCN = 0x0C; // disable watchdog timer WDTCN = 0x0D; delay=256; // Delay 1ms before polling XTFLD. while(delay--); while (!(OSCXCN & 0x80)); // Wait until external crystal has // started OSCXCN = 0x0C; // Switch to external oscillator OSCXCN = 0x88; // Disable internal oscillator; enable // missing clock detector. while (!(OSCXCN & 0x80)); // Wait until external crystal has </pre>	<pre> OSCICN = 0x08; // started. // Switch to external oscillator void init(void) { oscinit(); PRT2CF = 0x02; // enable P0.7 TMOD = 0x02; //Timer0 is 8-bit auto reload, Timer2 is 16 bit auto reload SET_TIMER(0,0xE8,0xA); // 0x10000h - 0x4118h = 0xBEE5h ; 2^16 - 1 - 16,667 = 48,869 IE = 0x82; //IE = 0x82; START_TIMER(0); // enable P0.7 (LED) as push-pull output PRT2CF = 0x80; //Must make sure not to pass a DC current through the transformer H_Out = 0; //if there is no DC blocking cap PRT2CF = 0x02; //enable P1.1 (H_Out) as push-pull output } void FcnFreq(void) interrupt 1 { /*-----*/ //FnFreq() times a square wave when the interrupt is properly set and not modified //Until its own termination. The square wave is outputable to a pin by toggling its //Register changing num_ints, TCON, IE Resources used: H_Out (P1.1), Timer0 Notes: Be sure not to pass a DC current through the transformer by initializing H_Out to 0 in the code and setting H_Out to 0 upon termination of the square wave output of FnFreq ***** */ if (num_ints < 8) { H_Out = ~H_Out; num_ints++; } else { H_Out=0; STOP_TIMER(0); SET_TIMER(0,0xE8,0xA); num_ints = 0; } } void BurstFreq(void) interrupt 5 { /*-----*/ //BurstFreq(), when T2 is set for autoreload, periodically restarts timer0 which will cause FnFreq //Q1) to start interrupting, causing a square wave to be generated at FnFreq()'s output again Registers changed: num_ints Resources used: LED, T2 Notes: Be sure not to pass a DC current through the transformer by initializing H_Out to 0 in the code and setting H_Out to 0 upon termination of the square wave output of FnFreq ***** */ START_TIMER(0); IE = 0x02; LED = ~LED; PRT2CF = 0x01; } void main(void) { init(); SPIinit(); while (1); } </pre>
---	--

Figure E.6 Microcontroller code for the ultrasound transmit board.

<pre> /* ultrasound_rx.c -- Cynal F206 code for an ultrasound rangefinding transmit board */ /* Written 8/1/03 by Dan Lovell for MIT Media Lab Gaitshoe Project Upon arrival of a rising edge on T2, the timer process is started. Arrival times and levels of various points of the incoming ultrasound wave are recorded and used to recreate the actual arrival time of the ultrasound ping Sysclk is 22.1148MHz -> one machine cycle = 45.2ns Speed of sound in air 346.65 m/s http://www.physicsclassroom.com/class/sound/popup.html Therefore an ultrasonic wave travels 0.00001566858 m in one machine cycle. The time it takes to travel from the transmitter to the receiver and back to the receiver is the comparator. Delay is on the order of us -> .0136inches One cycle of a 40KHz wave is 25us long. 25us * 346.65 m/s = 8.665m. Divide by 72 in 16 bit capture mode = 0.000296301 sec 0.000296301 seconds * 346.65 m/s = 0.07 m = 40 inches = 3' 4" therefore we in 16 bit capture mode should be sufficient to time all returns in seconds. start with 100ms. One LSB of the upper byte of TIMER2 = 2^8*.00001566858m = One cycle in time is .35 inches */ #include "206.h" #define temp PRT1IF bit LED = P0^0; bit T2EX = P0^1; bit T2INT = P0^7; int intCounter=0; int int2Counter=0; char charCounter=0; char time1L0; char time1L1; char time2L0; char time2L1; char T20w1IntFlag = T2CON^7; char T20w1IntFlag2 = T2CON^6; bit Trig2 = P1^6; bit NotTrig2 = P1^7; bit counter = 0; #define T2Capture() capturePin1; capturePin0; //SPI related declarations char ULTRASOUND_START_CODE = 0x96; char SPI0Data[4]; char ultrasound_data[4]; char ultrasound_data[0] = time1L1; char ultrasound_data[1] = time1L0; char ultrasound_data[2] = time2L1; char ultrasound_data[3] = time2L0; char SPI_BycCounter=0; char SPI_BycCounter2=0; char currConversion = 0; //used to keep track of where in ultrasound_data[] to write to void SPIInt(void) interrupt 6 { if(SPI0Data==ULTRASOUND_START_CODE){ T2EX = 0x01; //start the time of flight clock intCounter++; //increment the intCounter for first capture of threshold crossing int2Counter = 0; //reset the intCounter for first capture of threshold crossing SPI_BycCounter = 0; //reset the index into the SPI bytes to be sent via SPI SPI_BycCounter++; } if(SPI0Data==ULTRASOUND_START_CODE){ START_TIMER(2); //start the time of flight clock intCounter++; //increment the intCounter for first capture of threshold crossing int2Counter = 0; //reset the intCounter for first capture of threshold crossing SPI_BycCounter = 0; //reset the index into the SPI bytes to be sent via SPI SPI_BycCounter++; } SPI0=0; } void SPIInit(void) { //Setup interrupts TCON<=0x0A; //clear any interrupts waiting on INT0, INT1 PRT1IF<=0x00; IE = 0x010101001; //globally enables interrupts and INT0, INT1, T2, TO IE2 = 0x010; //enables software interrupts 6 and 7 (IE6, IE7) TCON<=0x05; //enable T2INT, INT0 to detect falling edge interrupts TMOD = 0x00; //TO is 16 bit counter/timer } void INT0Int(void) interrupt 0 { //This is triggered from J1 //Signal comes from J1 //This little snippet is to capture the first crossing of the threshold T2Capture(); // SET_TIMER(0,0xFF,0x80); // SPI0=0; if(intCounter>0) { ultrasound_data[0] = RCAP2L; ultrasound_data[1] = RCAP2H; } intCounter--; } void TOInt(void) interrupt 1 { //time delay between threshold crossing and A/D conversion to make sure //we have sampled STOP_TIMER(0); ADBUSY = 1; } void INT1Int(void) interrupt 2 { //This is triggered //Signal comes from J1 /LED1; } void T2Int(void) interrupt 5 { if (T2CapIntFlag){ //T2Capture(); T2CapIntFlag=0; } else { T20w1IntFlag=0; STOP_TIMER(2); SET_TIMER(2,0,0); } } void ADCInt(void) interrupt 15 { //Store the ADC value //Temporary put into ultrasound_data[2,3] for testing purposes ultrasound_data[2] = ADC0L; ultrasound_data[3] = ADC0H; ADCINT=0; } void EXInt(void) interrupt 18 { //This is Triggered //Signal comes from J2 T2Capture(); if(intCounter>0) { ultrasound_data[2] = RCAP2L; ultrasound_data[3] = RCAP2H; } } </pre>	<pre> IE = 0x80; //enable global interrupts SPI0CN = 0x01; //enable SPI in SLAVE mode SPI0CON = 0x00; //this sets SCK = SYCLK/20, maximum value is SYCLK/20 SPI0CSK = 0x09; //this sets SCK = SYCLK/20, maximum value is SYCLK/20 IE1 = 0x01; //Enable SPI interrupts PRT2MX = 0x01; //SPI pins SCK,MISO,MOSI,NSS available on P2 pins P2.0,P2.1,P2.2,P2.3 re spective PRT2CF = 0x02; //SCK,MOSI,NSS are input; MISO is output PRT2CF &= ~0x0D; } void ADCInit(void) { P1MODE = 0x08; // Input configuration for P1 P1_0 = GP_I/O; // Open-Drain Output/Input (Analog) P1_1 = GP_I/O; // Open-Drain Output/Input (Analog) P1_2 = GP_I/O; // Open-Drain Output/Input (Analog) } // ADC Configuration AMUXSL = 0x2A; // AMUX Channel Select Register ADC0CN = 0x00; // ADC Configuration Register ADC0CN = 0x00; // ADC Control Register ADC0N = 0x00; // ADC Data Word Register ADC0L = 0x00; // ADC Data Word Register REFCON = VDD; //Internal reference IE2 = 0x02; //Extended Interrupt Enable 2 } void oscinit(void) { int delay; OSCXCN = 0x06; // Enable external crystal WDTCN = 0x00; // disable watchdog timer WDTN = 0x00; delay=256; while(delay--); while (!OSCXCN & 0x80); // Delay >1 ms before polling XT1VLD. OSCICN = 0x0C; // Switch to external oscillator OSCICN = 0x0B; // Disable internal oscillator; enable // missing clock detector. while (!OSCXCN & 0x80); // Wait until external crystal has // started OSCICN = 0x0B; // Switch to external oscillator } void init(void) { oscinit(); // Setup I/O pins : comparator input and digital output PRT2MX = 0x80; //Disable T2EX, INT0 and INT1 available at port pins PRT2CF = 0x81; //make P0^7 PO1 and P0^0 (T2EX, PWM and LED) push-pull for output // Setup timer2 for capture GCON = 0x00; //all timers use raw clock SET_TIMER(2,0,0); //Initialize T2 TICON = 0x09; //Set T2 for capture on High to Low transition on T2EX (P0.7, pin12 on c </pre>
<pre> /* ultrasound_rx.c -- Cynal F206 code for an ultrasound rangefinding transmit board */ /* Written 8/1/03 by Dan Lovell for MIT Media Lab Gaitshoe Project Upon arrival of a rising edge on T2, the timer process is started. Arrival times and levels of various points of the incoming ultrasound wave are recorded and used to recreate the actual arrival time of the ultrasound ping Sysclk is 22.1148MHz -> one machine cycle = 45.2ns Speed of sound in air 346.65 m/s http://www.physicsclassroom.com/class/sound/popup.html Therefore an ultrasonic wave travels 0.00001566858 m in one machine cycle. The time it takes to travel from the transmitter to the receiver and back to the receiver is the comparator. Delay is on the order of us -> .0136inches One cycle of a 40KHz wave is 25us long. 25us * 346.65 m/s = 8.665m. Divide by 72 in 16 bit capture mode = 0.000296301 sec 0.000296301 seconds * 346.65 m/s = 0.07 m = 40 inches = 3' 4" therefore we in 16 bit capture mode should be sufficient to time all returns in seconds. start with 100ms. One LSB of the upper byte of TIMER2 = 2^8*.00001566858m = One cycle in time is .35 inches */ #include "206.h" #define temp PRT1IF bit LED = P0^0; bit T2EX = P0^1; bit T2INT = P0^7; int intCounter=0; int int2Counter=0; char charCounter=0; char time1L0; char time1L1; char time2L0; char time2L1; char T20w1IntFlag = T2CON^7; char T20w1IntFlag2 = T2CON^6; bit Trig2 = P1^6; bit NotTrig2 = P1^7; bit counter = 0; #define T2Capture() capturePin1; capturePin0; //SPI related declarations char ULTRASOUND_START_CODE = 0x96; char SPI0Data[4]; char ultrasound_data[4]; char ultrasound_data[0] = time1L1; char ultrasound_data[1] = time1L0; char ultrasound_data[2] = time2L1; char ultrasound_data[3] = time2L0; char SPI_BycCounter=0; char SPI_BycCounter2=0; char currConversion = 0; //used to keep track of where in ultrasound_data[] to write to void SPIInt(void) interrupt 6 { if(SPI0Data==ULTRASOUND_START_CODE){ T2EX = 0x01; //start the time of flight clock intCounter++; //increment the intCounter for first capture of threshold crossing int2Counter = 0; //reset the intCounter for first capture of threshold crossing SPI_BycCounter = 0; //reset the index into the SPI bytes to be sent via SPI SPI_BycCounter++; } if(SPI0Data==ULTRASOUND_START_CODE){ START_TIMER(2); //start the time of flight clock intCounter++; //increment the intCounter for first capture of threshold crossing int2Counter = 0; //reset the intCounter for first capture of threshold crossing SPI_BycCounter = 0; //reset the index into the SPI bytes to be sent via SPI SPI_BycCounter++; } SPI0=0; } void SPIInit(void) { //Setup interrupts TCON<=0x0A; //clear any interrupts waiting on INT0, INT1 PRT1IF<=0x00; IE = 0x010101001; //globally enables interrupts and INT0, INT1, T2, TO IE2 = 0x010; //enables software interrupts 6 and 7 (IE6, IE7) TCON<=0x05; //enable T2INT, INT0 to detect falling edge interrupts TMOD = 0x00; //TO is 16 bit counter/timer } void INT0Int(void) interrupt 0 { //This is triggered from J1 //Signal comes from J1 //This little snippet is to capture the first crossing of the threshold T2Capture(); // SET_TIMER(0,0xFF,0x80); // SPI0=0; if(intCounter>0) { ultrasound_data[0] = RCAP2L; ultrasound_data[1] = RCAP2H; } intCounter--; } void TOInt(void) interrupt 1 { //time delay between threshold crossing and A/D conversion to make sure //we have sampled STOP_TIMER(0); ADBUSY = 1; } void INT1Int(void) interrupt 2 { //This is triggered //Signal comes from J1 /LED1; } void T2Int(void) interrupt 5 { if (T2CapIntFlag){ //T2Capture(); T2CapIntFlag=0; } else { T20w1IntFlag=0; STOP_TIMER(2); SET_TIMER(2,0,0); } } void ADCInt(void) interrupt 15 { //Store the ADC value //Temporary put into ultrasound_data[2,3] for testing purposes ultrasound_data[2] = ADC0L; ultrasound_data[3] = ADC0H; ADCINT=0; } void EXInt(void) interrupt 18 { //This is Triggered //Signal comes from J2 T2Capture(); if(intCounter>0) { ultrasound_data[2] = RCAP2L; ultrasound_data[3] = RCAP2H; } } </pre>	<pre> intCounter=1; PRT1IF &= ~0x40; } void EXInt(void) interrupt 19 { //This is triggered //Signal comes from J2 T2Capture(); if(intCounter>0) { ultrasound_data[2] = RCAP2L; ultrasound_data[3] = RCAP2H; } intCounter--; } void main (void) { ADCInit(); int i; SPI0=0; LED=0; while(1) { //make sure the rising edge corresponds to the first arrival //since elsewhere the interrupt comes from the falling edge //This code is to catch the first rising edge of the J2 input //Temporarily disable the polling code while testing the ADC while(1) { if(~P0^1) { PRT1IF =0x40; counter++; break; } } while(1) { if(~Trig2) { break; } } } } </pre>
<pre> /* ultrasound_rx.c -- Cynal F206 code for an ultrasound rangefinding transmit board */ /* Written 8/1/03 by Dan Lovell for MIT Media Lab Gaitshoe Project Upon arrival of a rising edge on T2, the timer process is started. Arrival times and levels of various points of the incoming ultrasound wave are recorded and used to recreate the actual arrival time of the ultrasound ping Sysclk is 22.1148MHz -> one machine cycle = 45.2ns Speed of sound in air 346.65 m/s http://www.physicsclassroom.com/class/sound/popup.html Therefore an ultrasonic wave travels 0.00001566858 m in one machine cycle. The time it takes to travel from the transmitter to the receiver and back to the receiver is the comparator. Delay is on the order of us -> .0136inches One cycle of a 40KHz wave is 25us long. 25us * 346.65 m/s = 8.665m. Divide by 72 in 16 bit capture mode = 0.000296301 sec 0.000296301 seconds * 346.65 m/s = 0.07 m = 40 inches = 3' 4" therefore we in 16 bit capture mode should be sufficient to time all returns in seconds. start with 100ms. One LSB of the upper byte of TIMER2 = 2^8*.00001566858m = One cycle in time is .35 inches */ #include "206.h" #define temp PRT1IF bit LED = P0^0; bit T2EX = P0^1; bit T2INT = P0^7; int intCounter=0; int int2Counter=0; char charCounter=0; char time1L0; char time1L1; char time2L0; char time2L1; char T20w1IntFlag = T2CON^7; char T20w1IntFlag2 = T2CON^6; bit Trig2 = P1^6; bit NotTrig2 = P1^7; bit counter = 0; #define T2Capture() capturePin1; capturePin0; //SPI related declarations char ULTRASOUND_START_CODE = 0x96; char SPI0Data[4]; char ultrasound_data[4]; char ultrasound_data[0] = time1L1; char ultrasound_data[1] = time1L0; char ultrasound_data[2] = time2L1; char ultrasound_data[3] = time2L0; char SPI_BycCounter=0; char SPI_BycCounter2=0; char currConversion = 0; //used to keep track of where in ultrasound_data[] to write to void SPIInt(void) interrupt 6 { if(SPI0Data==ULTRASOUND_START_CODE){ T2EX = 0x01; //start the time of flight clock intCounter++; //increment the intCounter for first capture of threshold crossing int2Counter = 0; //reset the intCounter for first capture of threshold crossing SPI_BycCounter = 0; //reset the index into the SPI bytes to be sent via SPI SPI_BycCounter++; } if(SPI0Data==ULTRASOUND_START_CODE){ START_TIMER(2); //start the time of flight clock intCounter++; //increment the intCounter for first capture of threshold crossing int2Counter = 0; //reset the intCounter for first capture of threshold crossing SPI_BycCounter = 0; //reset the index into the SPI bytes to be sent via SPI SPI_BycCounter++; } SPI0=0; } void SPIInit(void) { //Setup interrupts TCON<=0x0A; //clear any interrupts waiting on INT0, INT1 PRT1IF<=0x00; IE = 0x010101001; //globally enables interrupts and INT0, INT1, T2, TO IE2 = 0x010; //enables software interrupts 6 and 7 (IE6, IE7) TCON<=0x05; //enable T2INT, INT0 to detect falling edge interrupts TMOD = 0x00; //TO is 16 bit counter/timer } void INT0Int(void) interrupt 0 { //This is triggered from J1 //Signal comes from J1 //This little snippet is to capture the first crossing of the threshold T2Capture(); // SET_TIMER(0,0xFF,0x80); // SPI0=0; if(intCounter>0) { ultrasound_data[0] = RCAP2L; ultrasound_data[1] = RCAP2H; } intCounter--; } void TOInt(void) interrupt 1 { //time delay between threshold crossing and A/D conversion to make sure //we have sampled STOP_TIMER(0); ADBUSY = 1; } void INT1Int(void) interrupt 2 { //This is triggered //Signal comes from J1 /LED1; } void T2Int(void) interrupt 5 { if (T2CapIntFlag){ //T2Capture(); T2CapIntFlag=0; } else { T20w1IntFlag=0; STOP_TIMER(2); SET_TIMER(2,0,0); } } void ADCInt(void) interrupt 15 { //Store the ADC value //Temporary put into ultrasound_data[2,3] for testing purposes ultrasound_data[2] = ADC0L; ultrasound_data[3] = ADC0H; ADCINT=0; } void EXInt(void) interrupt 18 { //This is Triggered //Signal comes from J2 T2Capture(); if(intCounter>0) { ultrasound_data[2] = RCAP2L; ultrasound_data[3] = RCAP2H; } } </pre>	<pre> intCounter=1; PRT1IF &= ~0x40; } void EXInt(void) interrupt 19 { //This is triggered //Signal comes from J2 T2Capture(); if(intCounter>0) { ultrasound_data[2] = RCAP2L; ultrasound_data[3] = RCAP2H; } intCounter--; } void main (void) { ADCInit(); int i; SPI0=0; LED=0; while(1) { //make sure the rising edge corresponds to the first arrival //since elsewhere the interrupt comes from the falling edge //This code is to catch the first rising edge of the J2 input //Temporarily disable the polling code while testing the ADC while(1) { if(~P0^1) { PRT1IF =0x40; counter++; break; } } while(1) { if(~Trig2) { break; } } } } </pre>

Figure E.7 Microcontroller code for the ultrasound receiver board.

E.3 Initial Results

The uncalibrated output from the two ultrasound sensors is shown in Figure E.8, along with the uncalibrated output of the four FSR sensors on both the right and left feet, to provide a reference to the gait cycle (data outliers caused by RF dropouts and undetected or mis-timed sonar burst were removed as described in Section 4.1.3).

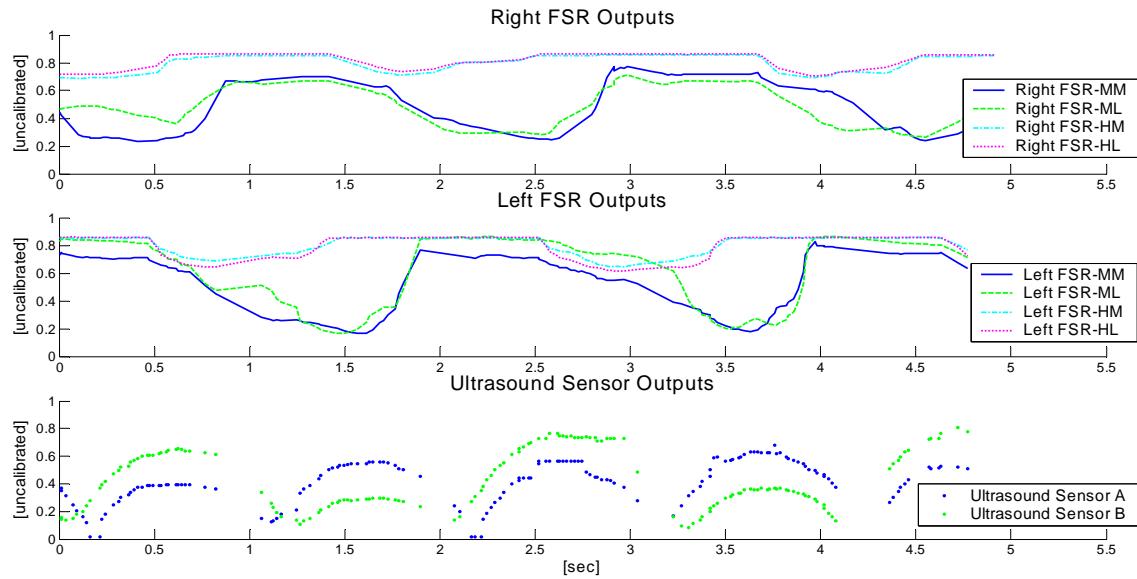


Figure E.8 Sample data from the two ultrasound sensors

The ultrasound receiver was located on the left foot, and the ultrasound transmitter was located on the right foot. The bottom graph demonstrates that the ultrasound sensor outputs are proportional to the distance between the receivers and the transmitter (ultrasound sensor A is located behind the stack, while ultrasound sensor B is located near the metatarsals). Shortly after 1 sec, the left foot is in mid-stance, and the right foot is in mid-swing, and as the transmitter on the right foot passes each of the receivers, the sensors reach local minimums in turn: the rear sensor (A) first, and then the front sensor (B), as each is passed by the transmitter. Then, shortly after 2 sec, the right foot is in mid-stance and the left foot is in mid-swing, and the sensors again reach local minimums, but in the opposite order, as this time the front sensor (B) passes the transmitter and is followed by the rear sensor (A).

Similarly, the local maximums can be observed shortly after 1.5 sec and 2.5 sec as one foot starts heel strike and the other foot starts toe off, corresponding to the maximum distance between the two feet.

This initial observation of the ultrasound sensor demonstrates that it may be a useful implementation on the GaitShoe for acquiring additional information about gait. As indicated in Figure 3.35, the two distances measured by the ultrasound receivers can be combined to determine the distance and the relative position between the two feet. However, before fully integrating the ultrasound sensor in the GaitShoe, both daughter boards should be redesigned. As seen in Figure E.1, the daughter boards (in particular, the receive daughter boards) are rather bulky and obtrusively located; now that the sensor has been demonstrated to work, a redesign of the positioning and fastening of these boards is necessary. The attachment also needs to be robust enough to withstand active gait. In addition, the typical power draw of the stack without the ultrasound sensor is 45 mA. On the stack with the transmit board the power draw increases to 65 mA, and on the stack with the receive board to 70 mA; this may restrict the use of the ultrasound sensor to applications where the batteries can be changed frequently.

Finally, a plot showing data from all of the sensors on both feet is shown in Figure E.8; the sensor outputs are uncalibrated (but staggered along the y-axis for legibility), during slow gait (to get the ultrasound data with as few outliers as possible). One of the ultrasound sensor outputs is plotted with the left foot data, and the other is plotted with the right foot data. This graph shows the full suite of sensor output of the GaitShoe.

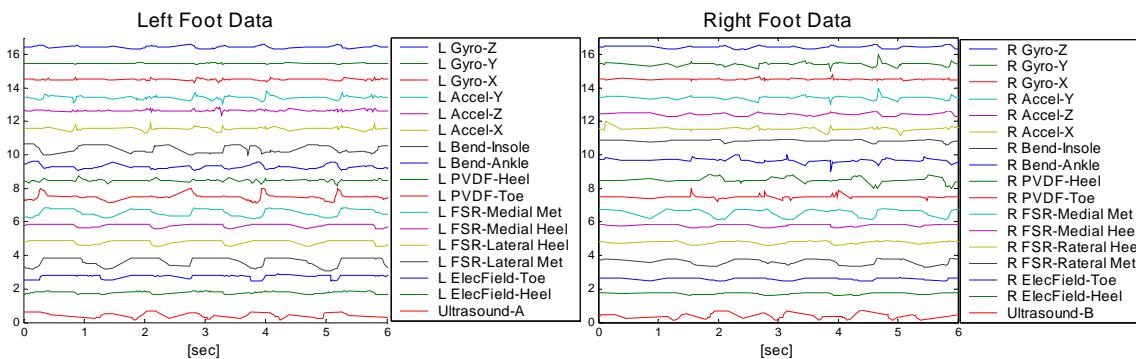


Figure E.9 Data collected from all sensors, during slow gait

Appendix F

CODE

F.1 Microcontroller Code

The code for the microcontrollers is based on code originally written by Ari Benbasat. It was modified by Steven Dan Lovell and the author. Figure F.1 contains the header file 206.h used by the microcontrollers, the basestation microcontroller code is in Figure F.2, and the stack microcontroller code, with comments indicating the lines which need to be switched for the left or right stack, is in Figure F.3 and Figure F.4.

```

/* 206.h -- Stub at a collection of useful macros for the */
/* ATmega128 microcontroller. */

/* Include gpt and sbit definitions */
#include <sys/8051.h>
#include <206SPR.h>

// Structures and unions
typedef struct {
    unsigned int word;
    unsigned char byte[2];
} lu;

// ADC Stuff
#define ADC_POR    ((P0&0x01)>>1) ADCHOL |= (P0&0x01)>>1;#define ADCDE L<-!(1<>pin);
#define SREG_ADC(un)    un.bye[1]=ADC5;un.lyte[0]=ADCON;
#define CLEAR_ADC(un)  ADC5=0x20;#define #MCKE |=1<>(pin);
#define ADC_POR(p0,pin) P0|=0x01;#define ADCDE |=1<>(pin);

/* Blocking send. Check to make sure that the previous send isn't still
going. Reset T1 pit. send. */
#define ADC_BLOCKING_SEND(ADCON,bitname)
/* Non-blocking send. Check if previous send is done, if so, send */
#define ADC_BLOCKING_SEND(ADCON,bitname)
#define ADC_PUT(x)

// Basic Timer Stuff
#define START_TIMER(num)    TH##num=1;
#define STOP_TIMER(num)     TH##num=0;
#define SHT_TIMER(num,high,low) T4num=low;TH##num=high;
#define SHT_TIMER2(num,high,low) RCA220num=RCA220high;RCA110num=low;
#define RELOAD_TIMER()    TH=RCAP28;T4=RCAL28

// More timer stuff
// This is not really the place for the PUTs, but there is no other way.
// Define the timer number, then call the macro
#define TIMER_RISE_TO_FALL(num,bitname,time) \
    SHT_TIMER(num,0x00,0x00); \
    STOP_TIMER(num); \
    PUT(0x55); \
    while(1){ if((timeout==0 || bitname==0)(break)); \
        if((timeout>0 || time>timeout)) \
            SHT_TIMER(num,0x00,0x00); \
        while(1){ if((timeout==1 || bitname==1)(break)); \
            if((timeout>0 || time>timeout)) \
                SHT_TIMER(num,0x00,0x00); \
            if((timeout==0 || bitname==0)(break)); \
            if((timeout>0 || time>timeout)) \
                SHT_TIMER(num,0x00,0x00); \
        } \
        SHT_TIMER(num,0x00,0x00); \
        while(bitname==1); \
        PUT(0x55); \
        while(bitname==0); \
        SHT_TIMER(num,0x00,0x00); \
        PUT(0x55); \
        while(bitname==1); \
    } \
    STOP_TIMER(num);

#define TIMER_RISE_TO_FALL_WAIT(num,bitname,time) \
    SHT_TIMER(num,0x00,0x00); \
    STOP_TIMER(num); \
    PUT(0x55); \
    while(1){ if((timeout==0 || bitname==0)(break)); \
        if((timeout>0 || time>timeout)) \
            SHT_TIMER(num,0x00,0x00); \
        while(1){ if((timeout==1 || bitname==1)(break)); \
            if((timeout>0 || time>timeout)) \
                SHT_TIMER(num,0x00,0x00); \
            if((timeout==0 || bitname==0)(break)); \
            if((timeout>0 || time>timeout)) \
                SHT_TIMER(num,0x00,0x00); \
        } \
        SHT_TIMER(num,0x00,0x00); \
        while(bitname==1); \
        PUT(0x55); \
        while(bitname==0); \
        SHT_TIMER(num,0x00,0x00); \
        PUT(0x55); \
        while(bitname==1); \
    } \
    STOP_TIMER(num);

```

Figure F.1 206.h Code

<pre> /* base.c -- Code for the basestation */ #include <stdio.h> #include "206.h" //Forward declarations void collectAnalog(void); void collectAccels(void); void collectTemp(void); void trans3Nibbles(unsigned char,unsigned char); void setchux(unsigned char int); #define SRT_REC CTR0<=1;fnable=0; #define SRT_TRANS CTR0<=0;fnable=1; // Declare ports char I2C_SDA = P3<1>; char I2C_SCL = P3<0>; char TMR0 = P2<7>; char TMR1 = P2<6>; char TMR2 = P2<5>; char TMR3 = P2<4>; char TMR4 = P2<3>; char TMR5 = P2<2>; char TMR6 = P2<1>; char TMR7 = P2<0>; //T2EX is improperly used in 206SF.h bit first; //Use this temp variable until that is resolved #define MUXDPORT 1 #define MUXPIN 7 // Mix stuff bit EM1 = P1<7>; bit AD0 = P1<2>; bit AD1 = P1<1>; bit AD2 = P1<0>; bit EN1 = P1<3>; bit EN2 = P1<4>; bit EN3 = P1<5>; bit EN4 = P1<6>; bit TMR8 = P0<7>; //T2EX is improperly used in 206SF.h // Variables lu gyrox; lu gyoy; lu gyoz; lu XTI; lu YTI; lu ZTI; bit in_cycle,pause,dirty; lu bendl,bnd2,far1,far2,far3,far4,far5,far6; lu checkedBit; //Used to check timestampCounter, capsense; unsigned char millow, milhigh; bit first,transFlag; void milliClock() interrupt 5 { } void oscinit(void) { int delay; //All the commented lines in oscinit were commented out for testing on //the cyano dev board //they should be uncommented when not testing on the dev board OSCXCN = 0x66; //Enable external crystal WDTCN = 0x0E; // disable watchdog timer WDTCN = 0xA0; } </pre>	<pre> delay255; while(delay--); // Delay >1 ms before polling XT1VLD. while (!!(OSCXCN & 0x80)); // Wait until external crystal has // started. OSCICN = 0x0C; // Switch to external oscillator OSCICN = 0x88; // Enable internal oscillator; enable // missing clock detector. while (!!(OSCXCN & 0x80)); // Wait until external crystal has // started. OSCICN = 0x88; // Switch to external oscillator // Setup function void init(void) { oscinit(); // Setup serial comm (using timer 1) PTROMX = UARTENABLE; // Turn on UART SCOM = 0x01; // Set SCLK instead of SVSCLK/12 for TI SCOM = USARTB1T RECEIVEON TRANSTIN; // Setup serial port TMOD = TIMER1_T1BBITRELOAD; // Set up serial timer and timestamp timer SET_TIMER(1, 0x0000, 0x0000); // Set up serial timer and timestamp timer .1184 MHz crystal ----- set baud rate here ----- T1 = 1; // Ready to send START_TIMER(1); //Setup timer for Timestamp CKCON = TODIV12; // Use SVSCLK/12 instead of SYCLK for T2 TMOD = T0THRM T0BBITRELOAD; // Set up timer 0 SET_TIMER(0, 0x0000, 0x0000); //Every TO INT is .ms if T0 uses SYCLK/12 START_TIMER(0); // Setup ADC ADCEN = 1; //Turn on ADC ADC0CON = 0x00; //Turn on the analog mux ADC1CON = 0x00; //Turn on the digital mux ADC0CF = SAR16[0]; //Set the ADC clock and prescaler REFSRC = 0x00; //Internal reference // Setup SPI PTROMX = 0x01; //Turn on SPI SPIONH = 0x03; //Set SPI to master mode SPICCR = 0x0F; //Set SPI Clock Rate //Setup T2 capture PTROMX = T2EX; //Turn on T2EX T2CON = T2TIMER T2CAPTURE T2ENABLE; EXEND = 1; // T2H:T2L -> RCAP2H:RCAP2L on SET_TIMER(2); // high->low transitions of T2EX //EI1 = 0x01; //Set RI and TI for use with interrupts, default to 0 RI=0; TI = 0; //Enable serial interrupt EO=1; //Enable Timer interrupt EA=1; //Make sure global interrupts are on // Inputs should be open drain (0) and high impedance (this is the default // for digital pins) or push-pull (1) and default low PTROCF = 0x01; //Turn on PTCF PTLCF = 0x7F; EN1=EN2=EN3=EN4=0; PTTF = 0x00; PTZCF = 0x00; //Turn on transmitter CTR1=1; CTR0=0; </pre>
<pre> //Blank variables gyro.word = gyro.yword = gyro.zword = 0; XTI.word = YTI.word = ZTI.word = DTI.word = 0; pause=in_cycle=dirty=0; void trans3Nibbles(unsigned char high, unsigned char low) { // DC Balance each six bit block unsigned char in,out; in = low&0xF; //first six bits out = lookUp[in]; SEND((out)<<4 high); //last four bits in = (low<<4 high<<4)&0xF; out = lookUp[in]; SEND((out)<<4 high); SEND_INT(out,transFlag); } void getTimestamp(void) { // Code for using T2 Capture T2EXComp = 0; //Normal since P2EX T2EXComp = 1; //Normal since P2EX goes through a high to low transition EO=1; //EO=1 means normally reset this flag after every T2 capture timestamp.byte[0] = RCAP2H; timestamp.byte[1] = RCAP2L; timestamp.word = timestampCounter.word; } void transmitSR() { // It should not be cleared here because SEND must see that TI has become one // or SEND will have been changed to work using interrupts instead of polling // Uses TMR8 for the purpose TI was used before transFlag = 1; TI = 0; } void receiveISR(void) { if(first & RI) { first = 0; getTimestamp(); } RI = 0; } void serial_int() interrupt 4 { transmitSR(); receiveISR(); } void T0_int() interrupt 1 { timestampCounter.word++; } void main(void) { int i; init(); SET_REC; while(1) { if(BDU1) { checkedBit.word += 1; } else { checkedBit.word = 0; } } //IF NOT USING ULTRASOUND, COMMENT FROM THE NEXT LINE } </pre>	<pre> for(i=0;i<14000;i++) // 14000, 11.2k (windows), 30000: 57.6k (mac) baudrate ----- // TI=1; //change if you change baud rate here // transmit ultrasound packet SET_TRANS; SEND_INT(0x0F,transFlag); SEND_INT(0x00,transFlag); SEND_INT(0x01,transFlag); SEND_INT(0x03,transFlag); SEND_INT(0x05,transFlag); SEND_INT(0x09,transFlag); SEND_INT(0x06,transFlag); SEND_INT(0x07,transFlag); SEND_INT(0x03,transFlag); SEND_INT(0x05,transFlag); SEND_INT(0x06,transFlag); // SET_REC; // //...REMOVED THIS LINE OUT (can leave in, but data rate will be lower) for(i=0;i<14000;i++) // 14000, 11.2k (windows), 30000: 57.6k (mac) baudrate ----- // change if you change baud rate here SET_TRANS; SEND_INT(0x0F,transFlag); SEND_INT(0x00,transFlag); SEND_INT(0x0C,transFlag); SEND_INT(0x01,transFlag); SEND_INT(0x03,transFlag); SEND_INT(0x05,transFlag); SEND_INT(0x09,transFlag); SEND_INT(0x06,transFlag); SEND_INT(0x07,transFlag); SEND_INT(0x03,transFlag); SEND_INT(0x05,transFlag); SEND_INT(0x06,transFlag); // while(1); first = 1; //Get timestamp when first gets set to zero in receiveISR SET_REC; for(i=0;i<14000;i++) // 14000, 11.2k (windows), 30000: 57.6k (mac) baudrate ----- // change if you change baud rate here TI=1; SET_TRANS; SEND_INT(0x0F,transFlag); SEND_INT(0x00,transFlag); SEND_INT(0x0C,transFlag); SEND_INT(0x01,transFlag); SEND_INT(0x03,transFlag); SEND_INT(0x05,transFlag); SEND_INT(0x09,transFlag); SEND_INT(0x06,transFlag); SEND_INT(0x07,transFlag); trans3Nibbles((checkedBit.byte[0],checkedBit.byte[1])); SEND_INT(0x03,transFlag); SEND_INT(0x06,transFlag); // while(1); first = 1; //Get timestamp when first gets set to zero in receiveISR SET_REC; } </pre>

Figure F.2 Basestation microcontroller code

```

/* stack.c -- code for the left and right stack */
#include <stdio.h>
#include "206.h"

//Forward declarations
void SPIInit(void);
void collectAnalog(void);
void collectAccels(void);
void transINBbles(unsigned char,unsigned char);
void setMux(unsigned char in);

#define SET_REC CTR0=1;TENable=0;
#define SET_TRANS CTR0=0;TENable=1;

// Define ports
abit CTR0 = P2$1;
abit CTR0 = P3$0;
abit TSEN0 = P2$7;
//#define SPWRENTE P2$3;
abit DIR1 = P2$4;
abit DIR2 = P2$5;
abit DIR3 = P2$6;
abit DIR5 = P3$6;
abit DIR6 = P3$2;

// SPI related declarations
#define SPI_RXNLSB P2$0
char SPI_flag=1; //used to indicate whether SPI transmission has concluded
char char_NSS=0;
char char_SCK=0;
char char_ULTRASOUND_START_CODE = 0x96;
char char_ULTRASOUND_DATA_CODE = -0x96;
char char_ULTRASOUND_DATA = 0xA0A4;
char char_ULTRASOUND_data[4];
char SPItransmitdata;
char SPItransmitdata,HavCounter, NoCounter;
char SPIInterrupted=1;
#define GARBLE 0
// Flag in location table
idata unsigned char lookup[64] = { (23, 27, 29, 39, 43, 45, 46, 51, 53, 54, 57,
58, 71, 75, 80, 83, 86, 89, 92, 95, 100, 103, 106, 109, 112, 114, 116,
135, 139, 141, 142, 147, 149, 150, 153,
172, 173, 174, 175, 176, 177, 178, 179,
177, 178, 180, 184, 199, 197, 198, 201, 202,
204, 209, 210, 212, 216, 225, 226, 228, 232)};

#define MEXPORT 1
#define SPI_RXNLSB 0
// More stuff
abit X = P1$7;
abit A0 = P1$2;
abit A1 = P1$1;
abit A2 = P1$0;
abit EN0 = P1$3;
abit EN2 = P1$4;
abit EN3 = P1$5;
abit EN4 = P1$6;
abit DATA = P2$0;
abit ASR = P2$1;
abit HAVE = P2$2;

// Variables
lu gyrox;
lu gyroy;
lu gyroz;
lu accx;
lu accy;
lu accelz;
lu accelz;

Registers changed:
SPIADEN, SPI_Counter
Resources used:
SPI - P2.0 (Ultrasonic_NSS), P2.0 (SCK), P2.1 (MISO), P2.2 (MOSI)
Notes:
NSS must be low for a few sysclk cycles before it is recognized as low by the slave
***** */

SPIInterrupted = 0;
UltraSound_start(); //signal to slave that a transmission is beginning
while(1+SPI_counter10); //wait for slave to receive the NSS_slave signal
SPI_Counter=0;
SPIData = SPI_Tx_BYTEx; //transmit to slave
}

void ultrasound_start() {
***** */
ultraSound_start(); //transmits a "start" command (one byte) via SPI to the ultrasound board
// This starts initializes the ultrasound range finding process on the ultrasound boards
Registers changed:
SPIADEN, SPI_Counter
Resources used:
SPI - P2.0 (Ultrasonic_NSS), P2.0 (SCK), P2.1 (MISO), P2.2 (MOSI)
Notes:
***** */

while((!SPIInterrupted))
{
    SPITransmit(ULTRASOUND_START_CODE); //transmit ultrasound start
}

void ultrasound_vals() {
***** */
ultraSound_vals(); //transmits a "data" command (one byte) via SPI to the ultrasound board
// This starts initializes the data transfer between the ultrasound receive board and the
// corresponding transmit board
Registers changed:
SPIIn, SPIADEN, SPI_Counter
Resources used:
SPI - P2.0 (Ultrasonic_NSS), P2.0 (SCK), P2.1 (MISO), P2.2 (MOSI)
Notes:
***** */

char SPIBytesCounter=0,SPIBytesToTransmit=4;
char char_NSS=0;
while(SPIInterrupted){
    SPITransmit(ULTRASOUND_DATA_CODE);
    SPITransmit(ULTRASOUND_DATA_CODE);
    while(SPIInterrupted);
    while(SPIBytesToTransmit - SPIBytesCounter != 0){
        SPITransmit(ULTRASOUND_DATA_CODE);
        while(SPIInterrupted);
        ultrasound_data(SPIBytesCounter++) = SPIODAT;
        SPIBytesCounter++;
        //This counter must be present so that the slave has ample time to write
        //Its byte to its SPI register. Failure to wait for atleast this long will
        //result in garbage being read by the master
        }
        counter=0;
    }

void oscinit(void)
{
    int delay;
    OSCKXN = 0x66; // Enable external crystal
    WDTCN = 0xD0; // disable watchdog timer
    WDTCN = 0xD0;
    delay=256;
    // Delay > ms before polling XTALD.
}

lu accelz2;
bit in_cycle.pause,dirty;
lu bndl,bndl2,fsl1,fsl2,fsl3,fsl4,fsl5,fsl6;
lu cap8, cap9, cap2, cap3;
lu timestamp,checkedbit;
unsigned char millow, milhigh;
unsigned short char counter;
int i=0; // used to create operations that allow me to set breakpoints
void millclock() interrupt 5
{
}

void SPIINT(c0d0) interrupt 6 {
***** */
SPIINT() occurs every time an SPI transmission finishes, causing an SPI interrupt.
Registers changed:
SPIPDR, SPIR_IN, counter
Resources used:
none
Notes:
SPI interrupts occur when an SPI byte has finished transmitting
SPI must be cleared by software in this interrupt or the MCU will hang at
the SPI interrupt vector (interrupt 6, memory location 0x0003).
***** */

//When an SPI interrupt occurs, an SPI byte has finished transmitting
UltraSound_MSSl: //raise slave's NSS pin so it can check its received byte
    SPIR_IN = 0x00; //clear the interrupt10); //wait for slave to receive the NSS_slave signal
    SPI_Counter=0;
    SPIPDR = SPIODAT;
    SPIInterrupted = 1;
    SPIPDR=0;
}

void SPIInit(void) {
***** */
SPIInit() initializes the MCU to run SPI interrupts
Registers changed:
SPIADEN, SPI_Counter
Resources used:
P2.0 (SCK), P2.1 (MISO), P2.2 (MOSI), P2.3 (NSS)
Notes:
if SPI is interrupted, it set then there should at the very least
be a "SPINT=0" statement in the interrupt to prevent the MCU from hanging
in the SPI interrupt vector (interrupt 6, memory location 0x0003).
***** */

// SPI is enabled as MASTER since SPIIN=1
IE |= 0x80; //enable global interrupt
SPIODAT = 0x03; //enable SPI in MASTER mode
SPIODC = 0x00; //set SCK to 1MHz, this is the default setting
SPILOCKR = 0x09; //this sets SCK = SYSCLK/20, maximum value is SYSCLK/20
EIE1 |= 0x01; //Enable SPI interrupts

PRT2MX |= 0x01; //SPI pins P2.0,P2.1,P2.2,P2.3 re
spectively
PRT2CF |= 0x05; //P2.0,P2.1,P2.2,P2.3 are Cygnal dev pins 25,26,23,24 respectively
PRT2CF |= 0x04; //SCK,MISO,MOSI,NSS are output; MISO,NSS are input
PRT3CF |= 0x04; //SCK,MISO,MOSI,NSS are output; P2.0,P2.1,P2.2,P2.3 re
spectively
PRT3CF |= 0x01; //SCK,MISO,MOSI,NSS are output; P2.0,P2.1,P2.2,P2.3 re
spectively
}

void SPITransmit(char SPI_TX_BYTE) {
***** */
SPITransmit sends a byte over the SPI bus
}

while(delay--);
while (!OSCXXN & 0x80));
// Wait until external crystal has
// started.
OSCICN = 0x0C; // Disable external oscillator
OSCICN = 0x88; // Enable external oscillator; enable
// missing clock detector.

while (!OSCXXN & 0x80));
// Wait until external crystal has
// started.
OSCICN = 0x08; // Switch to external oscillator
}

// Setup function
void init(void)
{
    oscinit();
    ASR = 6; // Set serial com (using timer 1)
    PTROM |= UARTENABLE; // Turn on UART
    CKCON = 0x00; // Use SYCLK, instead of SYSCLK/12
    TCON |= USARTBT; // RECEPTION; // Setup serial port
    TCON |= TTIME1 | TIBRELOAD; // Setup up serial timer
    ERT1 |= 0xF0; // CMRA=0xF0
    TMR1 |= 0x0000; // 57.6k (mac) baud timeout for 22.1184 MHz crystal <----- switch baud rate here ----->
    //TI = 1; // Ready to send
    START_TIMER1();
    // Setup ADC
    ADCEN = 1; // Turn on ADC
    ADMSK = 0x20; // Turn on the analog mux
    ADMT = 1; // Turn on tracking mode
    ADCOSC = 0x01|G1; //Set the ADC clock and prescaler
    REFCEN = 0; //Internal reference

    // Setup SPI
    PRT2MX |= 0x01; //P2.0,P2.1,P2.2,P2.3 set to master mode
    SPIODAT = 0x03; //Set SPI Clock Rate
    //Setup Accelerometer timer 0
    TMRD |= 0x0000 | T016BIT;
    //Setup timer 0 interrupt
    //T2CON |= T2TMR | T2RELOAD; // Reloading timer
    //ERT_TIMER2,D0x9,0x11); // Cycle start value is 0x0111 since each tick
    //ERT_TIMER2,D0x9,0x11); // 0x0111 * 5.614 = 0x06131 (0x0000-0x06131)x0.541us = 22ms
    //T2CON |= T2TMR | T2RELOAD; // Reloading timer
    //ERT_TIMER2,D0x8,0x07); // Cycle start value is 0x0111 since each tick
    //ERT_TIMER2,D0x8,0x07); // 0x0111 * 5.614 = 0x06131 (0x0000-0x06131)x0.541us = 22ms
    //System timer for timer 3
    //TMR3CN = 0x4d; // turn timer 3 on, divide by 12
    //TMR3RL = 0x47; // TMR3RL = 0x47;
    //TMR3RH = 0x07; // TMR3RH = 0x07;
    //Setup interrupts
    //ERT2 |= 1; // Timer 2 interrupt on
    EIE1 |= 0x01; // Serial port interrupt
    EIE1 |= 0x01; // Make sure global interrupts are on
    RI1=1; TI = 0; // Ready to receive

    // Inputs should be open drain (0) and high impedance (this is the default
    // Outputs should be push-pull (1) and default low
    PRT1CF = 0x0F; //EN1=EN2=EN3=EN4=0;
    PRT3CF = 0x47; //EN1=EN2=EN3=EN4=0;
    PRT2CF = 0x02;
    //Turn on transmitter
    CTRL1=1;
    CTRL0=0;
}

```

Figure F.3 Stack microcontroller code, part one

Figure F.4 Stack microcontroller code, part 2

The balanced byte code for wireless transmission is within both the basestation and the stack code. The basestation issues commands for each stack to send its complete packet of data for all the sensors in turn, and the basestation also includes a timestamp set by its on-board clock. In addition, during subject testing, it checked a pin connected to the BML TTL trigger, to detect when time zero was set on the BML system. These timing issues are summarized in Figure F.5.

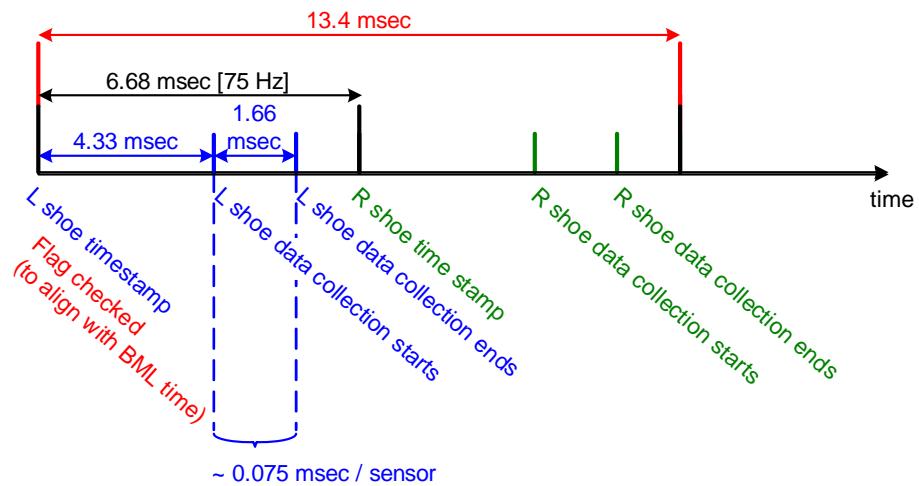


Figure F.5 Timing issues, as controlled by the basestation

F.2 Matlab Code

This section contains code for Matlab m-files used in this thesis; code located beneath the name of the file, and the order generally corresponds to the order in which these files were mentioned in the main text of the thesis. The color green is used for comments, red for text within the code, and the rest is black. To obtain these files electronically, email the author at <sjm@alum.mit.edu>.

timetrunc.m

<pre>C:\\$jm\Research\Matlab\THEESIS\timetrunc.m May 12, 2004 function [datatruncL,datatruncR,truncinfo]=timetrunc(vers, fig, saveit) %TIMETRUNC(version,fig,saveit) is used to truncate shoe data, using the trigger % signal. A window pops up allowing the user to select the data files of interest % (text files are expected). The data is graphed, and the user selects a % timepoint before the trigger. The program then truncates the data from % the start of the trigger, and re-graphs the data. The user can then % accept, or truncate the end. % % Version is used to locate the various data for plotting and the end of data. % This is so that this function can be easily updated if the data order is % changed or the data quantity increases. % % Version1: Basetime = 16, trigger = 17, computertime = 18. % Names are gvnNN where NNN is the three digit subject number. % One shoe file is expected. % % Version2: Basetime = 16, trigger = 17, computertime = 18. % Names are gvnNN where NNN is the three digit subject number. % Two shoe files are expected. GETSHOEADTOORDER 1 % % Version3: Basetime = 16, trigger = 17, computertime = 18. % Names are gvnNN where NNN is the three digit subject number. % Two shoe files are expected. GETSHOEADTOORDER 2 % % Version4: Basetime = 17, trigger = 18, computertime = 21. % Names are gvnNN where NNN is the three digit subject number. % Two shoe files are expected. GETSHOEADTOORDER ? % % % Fig is used to specify the figures the data will be plotted on (fig). % If no value is specified, the figures will be set to 1 and 2. % % Saveit is used to control whether the data is saved or not (if you just % want to look at data, you may not want to save it). If saveit=1, the output % will be saved to a mat file in c:\jm\Research\data\trunc_data and the mat % file will be named sjm101.mat. If saveit=2, the output will be saved to a mat % file in c:\jm\Research\data\trunc_data and the truncated % data will be named sjm102.mat. Saveit=0 or no value entered means the data % will not be saved. % % [running timetrunc with data which has previously been truncated will % cause the first version to be overwritten.] % % % (c)2003 Stacy J. Morris - sjm@alum.mit.edu % % ***** allargswho; [sizargs,col]=size(allargs); if sizargs<1 error('You must enter a version number. See help.'); elseif sizargs>2 error('That version number is non-existent!'); end</pre>	<pre>C:\\$jm\Research\Matlab\THEESIS\timetrunc.m May 12, 2004 function [fig1,% saveit=0; elseif sizargs<3 saveit=0; end if vers==1 vers==2 tpos=18; gyroz_pos=1; gyroy_pos=2; frrs_start=7; pvdvs_start_pos=11; trigger_pos=17; basetime_pos=16; namelength=6; if vers==1 splots=4; else splots=8; end interval=0.00668; elseif vers==3 tpos=19; gyroz_pos=1; gyroy_pos=2; frrs_start=11; pvdvs_start_pos=9; trigger_pos=17; basetime_pos=16; namelength=6; % splots=8; splots=7; interval=0.00668; elseif vers==4 tpos=21; gyroz_pos=1; gyroy_pos=2; frrs_start=11; pvdvs_start_pos=9; trigger_pos=18; basetime_pos=17; namelength=6; % splots=8; splots=7; interval=0.00668; else error('That version number is non-existent!'); end % ***** E F T ***** % This section prompts the user to select a (text) file, then extracts info % about the name for later saving. It also checks to make sure the data is % correct, then saves it to variable "rawdata". % % *****</pre>
<pre>C:\\$jm\Research\Matlab\THEESIS\timetrunc.m May 12, 2004 6:44:57 AM if vers==1 disp('Load shoe data.') % For data ease, in vers=1, data is just called left shoe dat a here elseif vers>2 disp('Load LEFT shoe data.') end [filename, fileloc] = uigetfile('.txt'); fileplace=[fileloc filename]; [toss,col]=size(filename); filename_abbrL=[filename(:,1:(colf-4)) 'tr']; %get rid of .txt (and preserves who ther L or R) filename_abbrL=[filename(:,1:namelength) '_trunc']; %use general identifies to store tr unc data filename_triggerinfo=[filename(:,1:(colf-5)) '_flag']; filenameR=[c:\\$jm\research\data\trunc_data\' filename_abbrL]; checker=exist(fileplace,'file'); if checker ~= 2 error('Shoetrunc: Invalid file name!'); end rawdata=dlmread(fileplace,' '); messageR=[filename ' loaded']; disp(messageR); if vers>1 vers==2 if vers==2 vers==3 fileplace=fileplace'; [toss,colR]=size(fileplaceR); fileplaceR(:,colf-4)=R'; filenameR=filename; filenameR=[filename(:,1:(colf-4)) 'R']; filename_abbrR=[filenameR(:,1:(colf-4)) 'tr']; %get rid of .txt filename_abbrR=[filenameR(:,1:namelength) '_trunc']; %use general identifies to sto re trunc data filenameR=[c:\\$jm\research\data\trunc_data\' filename_abbrR]; checker=exist(fileplaceR,'file'); if checker ~= 2 disp('Load RIGHT data.'); [filename, fileloc] = uigetfile('.txt'); fileplace=[fileloc filename]; [toss,colf]=size(filename); filename_abbrR=[filename(:,1:(colf-4)) 'tr']; %get rid of .txt filename_abbrR=[filename(:,1:namelength) '_trunc']; %use general identifies to store trunc data filenameR=[c:\\$jm\research\data\trunc_data\' filename_abbrR];</pre>	<pre>C:\\$jm\Research\Matlab\THEESIS\timetrunc.m May 12, 2004 6:44:57 AM Page 4 % checker=exist(fileplace,'file'); if checker ~= 2 error('Shoetrunc: Invalid file name!'); end fileplaceR=fileplace; rawdataR=dlmread(fileplaceR,' '); messageR=[filenameR ' loaded']; disp(messageR); % % This section plots three streams of data to the figure specified by fig % % ***** I G H T ***** time=timemodification(rawdata(:,tpos)); % time vector gyroz=rawdata(:,gyroz_pos); gyroy=rawdata(:,gyroy_pos); frrsL=rawdata(:,frrs_start:frrs_start+3); pvdvsL=rawdata(:,pvdvs_start_pos:pvdvs_start_pos+1); triggerL=rawdata(:,trigger_pos); if vers>2 trigspot=4; else trigspot=splots; end figure(fig); clf; dataplotterL(gyroz,timeL,fig,'raw data = left',splots,1,1,'b', 1) dataplotterL(gyroy,timeL,fig,'raw data = left',splots,1,1,'r', 1) dataplotterR(frrsL(:,1),timeL,fig,' ',splots,1,2,'b', 1) dataplotterR(frrsL(:,2),timeL,fig,' ',splots,1,2,'g', 1) dataplotterR(frrsL(:,3),timeL,fig,' ',splots,1,2,'g', 1) dataplotterR(frrsL(:,4),timeL,fig,' ',splots,1,2,'r', 1) dataplotterR(pvdvsL(:,1),timeL,fig,' ',splots,1,3,'b', 1) dataplotterR(pvdvsL(:,2),timeL,fig,' ',splots,1,3,'r', 1) % dataplotterR(triggerL,timeL,fig,' ',splots,1,trigspot,'b', 1) %removed rowL=length(timeL); % ***** I G H T ***** if vers>2 timeRtimemodecondition(rawdataR(:,tpos)); % time vector gyrozR=rawdataR(:,gyroz_pos); gyroyR=rawdataR(:,gyroy_pos); frrsR=rawdataR(:,frrs_start:frrs_start+3); pvdvsR=rawdataR(:,pvdvs_start_pos:pvdvs_start_pos+1); triggerR=rawdataR(:,trigger_pos); dataplotterR(gyrozR,timeR,fig,'raw data = right',splots,1,5-1,'b', 1) dataplotterR(gyroyR,timeR,fig,'raw data = right',splots,1,5-1,'r', 1) dataplotterR(frrsR(:,1),timeR,fig,' ',splots,1,6-1,'b', 1)</pre>

timetrunc.m, cont.

<pre>C:\sjm\Research\Matlab\THESIS\timetrunc.m May 12, 2004 dataplotterR(ffrsR(1,2),timeR,fig,'',splots,1,6-1,'k', 1) dataplotterR(ffrsR(1,3),timeR,fig,'',splots,1,6-1,'r', 1) dataplotterR(ffrsR(1,4),timeR,fig,'',splots,1,6-1,'g', 1) dataplotterR(fydfsh(1,2),timeR,fig,'',splots,1,7-1,'r', 1) dataplotterR(fydfsh(1,2),timeR,fig,'',splots,1,1,splots,'b', 1) rowR=length(timeR); end %%%%%%%%%%%%%%% % This section prompts the user to enter the start times - i.e. just before trigger time1=input('\n Where would you like the data to start?\n (pick a point just before subject starts walking)\n '); if time1==999;return;end time2=input('\n Where would you like the data to end?\n (pick a point just before subject turns)\n '); bottom=0.2; tops=1; iter1; while iter<50 if time2<=time1 time2=time1+10; end for i:splots subplot(splots,1,i) axis([time1 time2 bottom top]); end goodornot=input('\n Is this what you wanted? \n To change start time ONLY, enter 1, t o change end time ONLY, enter 2, to change both enter 3, \n and if done, enter 0 (10 to x com 10 to x com 10 to x);') if goodornot==0 if time1<timeL(1) (vers>2 & time1<timeR(1)) disp('Try again, start time must be after the beginning of both L and R times'); else if time2>timeL(1) (vers>2 & time2>timeR(1)) disp('Try again, end time must be before the end of both L and R times!'); else break end end else goodornot=1 time1=input('\n Where would you like the data to start?\n '); if vers>2 time2=input('\n Where would you like the data to end?\n '); else goodornot=3 time1=input('\n Where would you like the data to start?\n '); time2=input('\n Where would you like the data to end?\n '); elseif goodornot==10 bottom=0.1; tops=1; elseif goodornot==10 bottom=0.3; tops=0.7; end end iter=iter+1; end</pre>	<pre>C:\sjm\Research\Matlab\THESIS\timetrunc.m May 12, 2004 if goodornot==999;return;end % This section finds the start & end time points. startimept=findimepoints(time1,timeL,-1); endimept=findimepoints(time2,timeL,1); if vers>2 startimept=findimepoints(time1,timeR,-1); endimept=findimepoints(time2,timeR,1); end disp(''); disp('====='); disp(''); % This section finds the trigger point. iter1; bottom=1; top=1.2; while iter6 if iter==1 trigstart=timeL; trigend=timeR; else axis([time1 time2 bottom top]); time3=input(' Enter the time just before the trigger goes high. \n '); trigstart=im3+1.5; trigend=time1+1.5; end for i:splots subplot(splots,1,i) axis([trigstart trigend bottom top]); end t1=findimepoints(trigstart,timeL,-1); t2=findimepoints(trigend,timeL,1); [try,toss]=find(triggerR(t1:t2,1)==1); if vers>2 t1=findimepoints(trigstart,timeR,-1); t2=findimepoints(trigend,timeR,1); [try,toss]=find(triggerR(t1:t2,1)==1); end if (isempty(tryL)) & (vers>1 (vers>2 & not(isempty(tryR)))) subplot(splots,1,1); plot([timeL t1+tryL(1)-1],triggerR(t1+tryL(1)-1,1),'r.'); text(timeL+t1+tryL(1)-1,1,0.5,num2str(t1+tryL(1)-1)); else subplot(splots,1,splots); plot([timeR t1+tryR(1)-1],triggerR(t1+tryR(1)-1,1),'r.'); text(timeR+t1+tryR(1)-1,1,0.5,num2str(t1+tryR(1)-1)); end end</pre>
<pre>C:\sjm\Research\Matlab\THESIS\timetrunc.m May 12, 2004 if iter==1 goodornot=0; else goodornot=input('\n Is this what you wanted ? \n To find trigger points manua lly enter 1, else enter 0.\n '); end else goodornot=1; end if goodornot==0 trigstart=L1+tryL(1)-1; if vers>2 trigstartR=tR1+tryR(1)-1; end else disp('Click on trigger points until you find the first high trigger poi nt(s).'); disp('then hit escape.'); gname; trigstartL=input('\n Enter first high trigger point for L. \n '); if vers>2 trigstartR=input('\n Enter first high trigger point for R. \n '); end end ttl=trigstartL; if vers>2 ttlR=trigstartR; end if vers>3 rangeL=trigstartL-2:trigstartL+2; rangeR=trigstartR-2:trigstartR+2; dispdataL=[timeL(rangeL) rawdataL(rangeL,basetime_pos)/10000 triggerL(rangeL) one s(5,1) rangel'/10000 timeL(rangeR) rawdataL(rangeR,basetime_pos)/10000 triggerR(rangeR) ones(5,1)* 2 rangel'/10000]; dispdataR=[]; triginfo=sortrows(dispdataL,2); for i=1:10 if triginfo(i,3)==1 shoetrig=triginfo(i,4); linetrig=triginfo(i,5)*10000; autoid=[shoetrig linetrig]/10000; break end end elseif vers==1 rangeL=trigstartL-2:trigstartL+2; rangeR=trigstartR-2:trigstartR+2; dispdataL=[timeL(rangeL) rawdataL(rangeL,basetime_pos)/10000 triggerL(rangeL) one s(5,1) rangel'/10000]; triginfo=sortrows(dispdataL,2)</pre>	<pre>C:\sjm\Research\Matlab\THESIS\timetrunc.m May 12, 2004 for i=1:5 if triginfo(i,3)==1 shoetrig=triginfo(i,4); linetrig=triginfo(i,5)*10000; autoid=[shoetrig linetrig]/10000; break end end disp(''); autoides=input('\n Do you want to accept the autoid numbers? Enter 0 is yes, 1 to co nfirm. \n '); if autoides==0;break;end shoetrig=input('\n Enter 1 if trigger high occurs first in L, 2 if first in R.\n '); if shoetrig==1 linetrig=input('\n Enter line number of first trigger high.\n '); disp(''); netrig='You entered ' num2str(shoetrig) ' for shoetrig, and ' num2str(lin etrig) ' for linetrig.';disp(netrig); goodornot=input('\n Is this what you wanted ? \n To iterate enter 1, else if done, en ter 0.\n '); if goodornot==0 break end iter=iter+1; end dataruncl=rawdataL(startimept:endimeptL,:);pos; if vers>2 dataruncl=rawdataR(startimeptR:endimeptR,:);pos; end if shoetrig==1; truncinfo=[shoetrig linetrig-startimeptL+1]; else truncinfo=[shoetrig linetrig-startimeptR+1]; end if vers>1 append(matfilenameL,dataruncl,filename_abbr1L); append(matfilenameR,dataruncl,filename_abbr1R); message(['** filename_abbr1 ' saved.']); disp(''); disp(message); if vers>2 append(matfilenameL,dataruncl,filename_abbr1R); message(['** filename_abbr1 ' saved.']); disp(''); disp(message); end end</pre>

getshoedataorder.m

```
C:\sjm\Research\Matlab\THEESIS\getshoedataorder.m      Page 1
May 12, 2004                                         6:51:59 AM
function [shoedata_ret1, shoedata_ret2, qualinfo1, qualinfo2]=getshoedataorder(subjvers, %
shoedata1, shoedata2, triggerflag, fig)
%GETSHOEDATAORDER(subjvers, shoedata1, shoedata2, flagdata, fig) takes in shoe data, and %
returns the data in a revised %
reorder, with the time adjusted. Data is linearly interpolated at bad-data pts and during %
time gaps with 3 or %
fewer packets missing. Use the version number to specify the type of shoe data. If two %
shoes, shoedata1 must be Left shoe, int_or_shoedata2 must be Right shoe.
%
% Data is returned in this reorder:
%   1 fsr_mm, 2 fsr_hl, 3 fsr_ml, 4 fsr_ml, 5 pvdif_h, 6 pvdif_t, 7 bend_i, 8 bend_a,
%   9 gyroz, 10 gyrox, 11 gyroy, 12 accely, 13 gyrox, 14 accelx,
%   15 capacitiveheel, 16 capacitive-toe,
%   17 - future use, 18 - future use, 19 - future use, 20 - future use
%   21 time, 22 info about replaced data
%
% Subvers:
% -----
% Version 1: Subjects 011-019
% -----
% o L - 18 columns in this reorder:
%   gyroz, gyrox, gyrox, accely, accelx, fsr_hl, fsr_ml, fsr_mm, fsr_hm,
%   pvdif_h, pvdif_t, bend_i, bend_a, gyroz, gyrox, gyrox, baseetime, trigger, computertime
%   -bend_i needs to be flipped for right
%   (-no fsr_hi for 18, 17)
% o R - 18 columns in this reorder:
%   gyroz, gyrox, gyrox, accely, accelx, fsr_hl, fsr_ml, fsr_mm, fsr_hm,
%   pvdif_t, pvdif_h, bend_i, bend_a, gyroz, gyrox, gyrox, baseetime, trigger, computertime
%   -11 bend funky ...
%
% -----
% Version 2: Subjects 020-026 (cap data: 022-026) - all new transmit, hence, new pvdif
% -----
% o L - 18 columns in this reorder:
%   gyroz, gyrox, gyrox, accely, accelx, bend_i, bend_a, pvdif_h, pvdif_t,
%   fsr_mm, fsr_hm, fsr_hl, fsr_ml, cap_h, baseetime, trigger, computertime
% o R - 18 columns in this reorder:
%   gyroz, gyrox, gyrox, accely, accelx, bend_i, bend_a, pvdif_t, pvdif_h,
%   fsr_mm, fsr_hm, fsr_hl, fsr_ml, cap_h, baseetime, trigger, computertime
%
if nargin<5; fig=0;end
interval=.00668;
if subjvers==1
    reorder1=[9 11 13 12 10 14 2 4 3 1 5 6 7 8 0 15]; %left
    reorder2=[9 11 13 12 10 14 2 4 3 1 6 5 7 8 0 15]; %right
    [datalength1,datawidth1]=size(shoedata1);
    if notisempty(shoedata1) & datawidth1!=18
        error('Data is wrong size!')
    else
        basetime1=shoedata1(:,16);
        trigger1=shoedata1(:,17);
        comptime1=shoedata1(:,18);
    end
    dataamount=15;
elseif subjvers==2
    reorder1=[9 11 13 12 10 14 7 8 5 6 3 1 2 4 15 16]; %left
    reorder2=[9 11 13 12 10 14 7 8 6 5 3 1 2 4 15 16]; %right
    [datalength1,datawidth1]=size(shoedata1);
    if notisempty(shoedata1) & datawidth1!=18
        error('Data is wrong size!')
    else
        basetime1=shoedata1(:,16);
        trigger1=shoedata1(:,17);
        comptime1=shoedata1(:,18);
    end
    dataamount=15;
else
    error('please enter a valid version number')
end
%%Condition Time!!!!!!!
if subjvers==1 | subjvers==2
    [timel,time2,triggap]=timedadjuster([basetime1 trigger1 comptime1],[basetime2 trigger2 comptime2],triggerflag);
end
%% Re-reorder and "smooth" data %%%%%%
if notisempty(shoedata1)
    [shoedata_ret1,qualinfo1]=dataadjuster([shoedata1(:,1:dataamount) timel],reorder1,fig)
else
    dataadj21=[];
end
if notisempty(shoedata2)
    %% quality info %%
    qualinfo2(1,3:4)=[1 triggap];
    qualinfo2(1,3:4)=[2 triggap];
end
```

```
C:\sjm\Research\Matlab\THEESIS\getshoedataorder.m      Page 2
May 12, 2004                                         6:51:59 AM
function [shoedata_ret1, shoedata_ret2, qualinfo1, qualinfo2]=getshoedataorder(subjvers, %
shoedata1, shoedata2, triggerflag, fig)
%GETSHOEDATAORDER(subjvers, shoedata1, shoedata2, flagdata, fig) takes in shoe data, and %
returns the data in a revised %
reorder, with the time adjusted. Data is linearly interpolated at bad-data pts and during %
time gaps with 3 or %
fewer packets missing. Use the version number to specify the type of shoe data. If two %
shoes, shoedata1 must be Left shoe, int_or_shoedata2 must be Right shoe.
%
% Data is returned in this reorder:
%   1 fsr_mm, 2 fsr_hl, 3 fsr_ml, 4 fsr_ml, 5 pvdif_h, 6 pvdif_t, 7 bend_i, 8 bend_a,
%   9 gyroz, 10 gyrox, 11 gyroy, 12 accely, 13 gyrox, 14 accelx,
%   15 capacitiveheel, 16 capacitive-toe,
%   17 - future use, 18 - future use, 19 - future use, 20 - future use
%   21 time, 22 info about replaced data
%
% Subvers:
% -----
% Version 1: Subjects 011-019
% -----
% o L - 18 columns in this reorder:
%   gyroz, gyrox, gyrox, accely, accelx, fsr_hl, fsr_ml, fsr_mm, fsr_hm,
%   pvdif_h, pvdif_t, bend_i, bend_a, gyroz, gyrox, gyrox, baseetime, trigger, computertime
%   -bend_i needs to be flipped for right
%   (-no fsr_hi for 18, 17)
% o R - 18 columns in this reorder:
%   gyroz, gyrox, gyrox, accely, accelx, fsr_hl, fsr_ml, fsr_mm, fsr_hm,
%   pvdif_t, pvdif_h, bend_i, bend_a, gyroz, gyrox, gyrox, baseetime, trigger, computertime
%   -11 bend funky ...
%
% -----
% Version 2: Subjects 020-026 (cap data: 022-026) - all new transmit, hence, new pvdif
% -----
% o L - 18 columns in this reorder:
%   gyroz, gyrox, gyrox, accely, accelx, bend_i, bend_a, pvdif_h, pvdif_t,
%   fsr_mm, fsr_hm, fsr_hl, fsr_ml, cap_h, baseetime, trigger, computertime
% o R - 18 columns in this reorder:
%   gyroz, gyrox, gyrox, accely, accelx, bend_i, bend_a, pvdif_t, pvdif_h,
%   fsr_mm, fsr_hm, fsr_hl, fsr_ml, cap_h, baseetime, trigger, computertime
%
if nargin<5; fig=0;end
interval=.00668;
if subjvers==1
    reorder1=[9 11 13 12 10 14 7 8 5 6 3 1 2 4 15 16]; %left
    reorder2=[9 11 13 12 10 14 7 8 6 5 3 1 2 4 15 16]; %right
    [datalength2,datawidth2]=size(shoedata2);
    if notisempty(shoedata2) & datawidth2!=18
        error('Data is wrong size!')
    else
        basetime2=shoedata2(:,16);
        trigger2=shoedata2(:,17);
        comptime2=shoedata2(:,18);
    end
    dataamount=15;
elseif subjvers==2
    reorder1=[9 11 13 12 10 14 7 8 5 6 3 1 2 4 15 16]; %left
    reorder2=[9 11 13 12 10 14 7 8 6 5 3 1 2 4 15 16]; %right
    [datalength2,datawidth2]=size(shoedata2);
    if notisempty(shoedata2) & datawidth2!=18
        error('Data is wrong size!')
    else
        basetime2=shoedata2(:,16);
        trigger2=shoedata2(:,17);
        comptime2=shoedata2(:,18);
    end
    dataamount=15;
else
    error('please enter a valid version number')
end
%%Condition Time!!!!!!!
if subjvers==1 | subjvers==2
    [timel,time2,triggap]=timedadjuster([basetime1 trigger1 comptime1],[basetime2 trigger2 comptime2],triggerflag);
end
%% Re-reorder and "smooth" data %%%%%%
if notisempty(shoedata1)
    [shoedata_ret1,qualinfo1]=dataadjuster([shoedata1(:,1:dataamount) timel],reorder1,fig)
else
    dataadj21=[];
end
if notisempty(shoedata2)
    %% quality info %%
    qualinfo2(1,3:4)=[1 triggap];
    qualinfo2(1,3:4)=[2 triggap];
end
```

```
C:\sjm\Research\Matlab\THEESIS\getshoedataorder.m      Page 3
May 12, 2004                                         6:51:59 AM
if fig==0
    fig=6;
end
[shoedata_ret1,qualinfo2]=dataadjuster([shoedata2(:,1:dataamount) time2],reorder2,fig)
+6);
else
    dataadj21=[];
end

%%% quality info %%
qualinfo2(1,3:4)=[1 triggap];
qualinfo2(1,3:4)=[2 triggap];
```

timeadjuster.m

timeadjuster.m, cont.

<pre>C:\sjm\Research\Matlab\THESIS\timeadjuster.m May 12, 2004 Page 5 7:03:20 AM bs((datainfo_back(i,4)<4096)-datainfo_back(i,5)<0 disp('duplicating!'); %remove if this never occurs at1Ns(i,:) [datainfo_back_nxro abs((datainfo_back(i,3)+n*.4096)-datainfo_back(i,5))] % at1Ns(i,4)==100 else at1Ns(i,4)=0; at10ss=at10ss+at1Ns(i,:); removers=[removers;i];%can't remove now or will mess up rest of loop end end end if notisempty(removers) at1Ns(removers,:)=[]; at1NsL=at1NsL-length(removers); end %at1Ns(i,1)=0; %set time in row 1 = 0 (will be adjusted later, if triggerflag provided) for i=2:at1NsL if at1Ns(i,4)<0 n=at1Ns(i,:)/(-50); end if at1Ns(i,4)==1 at1Ns(i,7)=at1Ns(i-1,7)+at1Ns(i,1)-at1Ns(i-1,1); if at1Ns(i,7)<at1Ns(i-1,7) %missed rollover (can happen if many packets marked as 0) n=floor((at1Ns(i,3)-at1Ns(i-1,3))/0.4096); if at1Ns(i,1)-at1Ns(i-1,1)<0 n=n+1; end at1Ns(i,7)=at1Ns(i-1,7)+at1Ns(i,1)+n*.4096; at1Ns(i,4)=i-1; %change to -1 in to indicate missed rollover end else if at1Ns(i,4)<0 at1Ns(i,7)=at1Ns(i-1,7)+at1Ns(i,1)+n*.4096; elseif at1Ns(i,4)==0 disp('fix this! (shouldn''t happen) - 3 - timeadj'); at1Ns(i,:); end end end % need to reconstruct full matrix; add at10ss back in at1Ns=at1Ns(at1NsL); at10L=toss(size(at10ss)); %get rid of "fake" missing shoe data if oneshoming==0 at10ss=at10s(2:at10L,:); C:\sjm\Research\Matlab\THESIS\timeadjuster.m May 12, 2004 Page 6 7:03:20 AM % potentially bad, includes last rows earlierstart=last(CTprevious)-10; if earlierstart<1 earlierstart=1; end r2f=last(CTprevious)-earlierstart + 2; %+1 for loc of last(CTprevious), + 1 for loc of rowfix else earlierstart=last(CTprevious); r2f=2; end fixedrowsmta_fix([atir(earlierstart:last(CTprevious),:) at10ss(i,:)]; atir(CTfollowing(1,:),r2f)); %use nta_fix atir([atir(earlierstart-1,:);fixedrows;atir(CTfollowing(1)+1:atirL,:)]; atir(CTfollowing(1)+1:atirL,:)); else if isempty(CTprevious) %at10ss(i,:) if at10ss(i,:).is at the beginning of atir if CTfollowing(1)==1 at10ss(i,:).has a unique CT & is the new row 1 fixrows([at10ss(i,:); 0; atir(1,:)]; fixrows(2,i)-50); %set to -50 so okay if a rollover atir(1,:)=atir(1,:)+50; %fix first row, and nta_fix will look for a -50 fixedrowsmta_fix(fixrows,1); else if at10ss(i,:).has a repeat-CT r2f=last(CTprevious)-earlierstart + 2 atir(1,9)-5 atir(1,7)-5; atir (1:CTfollowing(1),r2f); %use nta_fix with fake first row reorderedrows=reorderdrows(2:CTfollowing(1)+2,:); atir=[reorderdrows;atir(CTfollowing(1)+1:atirL,:)]; atirL=atirL+1; end end else %empty(CTfollowing) => at10ss(i,:).is at the end of atir if last(CTprevious).is at the end of atir if at10ss(i,:).has a unique CT & is the new end row if atir(last(CTprevious),4)==0 %if previous row C:\sjm\Research\Matlab\THESIS\timeadjuster.m May 12, 2004 Page 7 7:03:20 AM % potentially bad, includes last rows earlierstart=last(CTprevious)-10; if earlierstart<1 earlierstart=1; end r2f=last(CTprevious)-earlierstart + 2; %+1 for loc of last(CTprevious), + 1 for loc of rowfix else earlierstart=last(CTprevious); r2f=2; end fixedrowsmta_fix([atir(earlierstart:last(CTprevious),:) at10ss(i,:)]; atir(CTfollowing(1,:),r2f)); %use nta_fix_rr with fake last row reorderdrows=reorderdrows(1:atir-last(CTprevious)+1,:); atir=atir(1:last(CTprevious)-1,:);reorderedrows; atirL=atirL+1; end end %checks Dtime2=(atir(atirL,3)-atir(atirL,2)) - (atir(atirL,7)-atir(1,7)); if Dtime2>0.1 Dtime2<0.3 disp('big time gap?') Dtime1 Dtime2 end Dtimes=diff(atir(:,7)); Dflag1; for i=1:length(Dtimes) if Dtimes(i)<.006 if Dflag1 disp('check time generation in timeadj'); Dflag=0; end [1 Dtmes(i) atir(i,5:6) atir(i+1,5:6)] end end if length(atir)-atirL disp('check length of atir wrt atirL in timeadj') end if length(atir)=length+lengthR disp('check length of atir wrt length+lengthR in timeadj') end % fix with trigger </pre>	<pre>C:\sjm\Research\Matlab\THESIS\timeadjuster.m May 12, 2004 Page 6 7:03:20 AM at10L=at10L-1; end if notisempty(at10ss) %need to reorder at10ss so we go consecutively at10s=sortbyoriginalindex(sortrows(at10s,6)); at10s=sortrows(at10s,sortbyoriginalindex,3); end % at10s(:,2)=at10s(:,1)/1000 for i=1:at1NsL CPrevious=find(atir(:,3)>at10ss(i,3)); CTfollowing=find(atir(:,3)>at10ss(i,3)); if notisempty(CTprevious) & notisempty(CTfollowing) %at10ss(i,:) is in the middle of atir if last(CTprevious)>CTfollowing(1) %at10ss(i,:). has a unique CT if atir(last(CTprevious),2)>1 atir(last(CTprevious),4)==0 %if previous row is potentially bad, include more rows earlierstart=last(CTprevious)-10; if earlierstart<1 earlierstart=1; end r2f=last(CTprevious)-earlierstart + 2; %+1 for loc of last(CTprevious), + 1 for loc of rowfix else earlierstart=last(CTprevious); r2f=2; end fixedrowsmta_fix([atir(earlierstart:last(CTprevious),:) at10ss(i,:)]; atir(CTfollowing(1:i,:),r2f)); %use nta_fix atir([atir(earlierstart-1,:);fixedrows;atir(CTfollowing(1)+1:atirL,:)]; atir(CTfollowing(1)+1:atirL,:)); else if isempty(CTprevious) %at10ss(i,:). is at the beginning of atir if CTfollowing(1)==1 at10ss(i,:).has a unique CT & is the new row 1 fixrows([at10ss(i,:); 0; atir(1,:)]; fixrows(2,i)-50); %set to -50 so okay if a rollover atir(1,:)=atir(1,:)+50; %fix first row, and nta_fix will look for a -50 fixedrowsmta_fix(fixrows,1); else if at10ss(i,:).has a repeat-CT r2f=last(CTprevious)-earlierstart + 2 atir(1,9)-5 atir(1,7)-5; atir (1:CTfollowing(1),r2f); %use nta_fix with fake first row reorderedrows=reorderdrows(2:CTfollowing(1)+2,:); atir=[reorderdrows;atir(CTfollowing(1)+1:atirL,:)]; atirL=atirL+1; end end else %empty(CTfollowing) => at10ss(i,:).is at the end of atir if last(CTprevious).is at the end of atir if at10ss(i,:).has a unique CT & is the new end row if atir(last(CTprevious),4)==0 %if previous row C:\sjm\Research\Matlab\THESIS\timeadjuster.m May 12, 2004 Page 8 7:03:20 AM if notisempty(triggerflag) trigger=atir(:,2)-1; if isempty(trigger) length(trigger)>1 error('problem with finding triggerflag') end if atir(trigger,5)=1 %if left shoe timetosubtract=atir(trigger,7); else timetosubtract=atir(trigger,7)-int; end atir(:,8)=atir(:,7)-timetosubtract; atir=atir(1:last(CTprevious)+1,:); atirL=atirL+1; dropgap=(atir(1:last(CTprevious)+1,:)-atir(1,:))/0.0134-1; %number of dropped packets else %no trigger provided then just set time in first row = 0 (if it isn't already - i.e. if atir(1,:).is the real first row) atir(:,8)=0; atir(:,8)=atir(:,7)+atir(1,7); if atir(:,7)>0 disp('though this shouldn''t happen: timeadj no trig') atir(:,8)=atir(:,7)-atir(1,7); else atir(:,8)=atir(:,7); end triggergap[]; %triggergap irrelevant end %un-sort atir to recover L and R time data unsortatir=sortrows(atir,5); %sort by shoe type [fr,tos]=find(unsortatir(:,5)==2); timetemp=sortatir(1:fr-1,:); timetemp=sortatir(fr:atirL,:); timetemp=sortatir(1:fr-1,:); timetemp=sortatir(fr:atirL,:); %check that they are in the right order! Lcheck=timetemp(:,6)-btr1(:,6);Rcheck=timetemp(:,6)-btr2(:,6); LcheckNz=find(Lcheck==0);RcheckNz=find(Rcheck==0); if not isempty(LcheckNz)disp('error in reconstructing timeL');end if not isempty(RcheckNz)disp('error in reconstructing timeR');end timeL=timetemp(:,8);timeR=timetemp(:,8); </pre>
<pre>C:\sjm\Research\Matlab\THESIS\timeadjuster.m May 12, 2004 Page 7 7:03:20 AM is potentially bad, includes last rows earlierstart=last(CTprevious)-10; if earlierstart<1 earlierstart=1; end r2f=last(CTprevious)-earlierstart + 2; %+1 for loc of last(CTprevious), + 1 for loc of rowfix else earlierstart=last(CTprevious); r2f=2; end fixedrowsmta_fix([atir(earlierstart:last(CTprevious),:) at10ss(i,:)]; atir(CTfollowing(1,:),r2f)); %use nta_fix_rr with fake last row reorderdrows=reorderdrows(1:atir-last(CTprevious)+1,:); atir=atir(1:last(CTprevious)-1,:);reorderedrows; atirL=atirL+1; end end %checks Dtime2=(atir(atirL,3)-atir(atirL,2)) - (atir(atirL,7)-atir(1,7)); if Dtime2>0.1 Dtime2<0.3 disp('big time gap?') Dtime1 Dtime2 end Dtimes=diff(atir(:,7)); Dflag1; for i=1:length(Dtimes) if Dtimes(i)<.006 if Dflag1 disp('check time generation in timeadj'); Dflag=0; end [1 Dtmes(i) atir(i,5:6) atir(i+1,5:6)] end end if length(atir)-atirL disp('check length of atir wrt atirL in timeadj') end if length(atir)=length+lengthR disp('check length of atir wrt length+lengthR in timeadj') end % fix with trigger </pre>	<pre>C:\sjm\Research\Matlab\THESIS\timeadjuster.m May 12, 2004 Page 8 7:03:20 AM if notisempty(triggerflag) trigger=atir(:,2)-1; if isempty(trigger) length(trigger)>1 error('problem with finding triggerflag') end if atir(trigger,5)=1 %if left shoe timetosubtract=atir(trigger,7); else timetosubtract=atir(trigger,7)-int; end atir(:,8)=atir(:,7)-timetosubtract; atir=atir(1:last(CTprevious)+1,:); atirL=atirL+1; dropgap=(atir(1:last(CTprevious)+1,:)-atir(1,:))/0.0134-1; %number of dropped packets else %no trigger provided then just set time in first row = 0 (if it isn't already - i.e. if atir(1,:).is the real first row) atir(:,8)=0; atir(:,8)=atir(:,7)+atir(1,7); if atir(:,7)>0 disp('though this shouldn''t happen: timeadj no trig') atir(:,8)=atir(:,7)-atir(1,7); else atir(:,8)=atir(:,7); end triggergap[]; %triggergap irrelevant end %un-sort atir to recover L and R time data unsortatir=sortrows(atir,5); %sort by shoe type [fr,tos]=find(unsortatir(:,5)==2); timetemp=sortatir(1:fr-1,:); timetemp=sortatir(fr:atirL,:); timetemp=sortatir(1:fr-1,:); timetemp=sortatir(fr:atirL,:); %check that they are in the right order! Lcheck=timetemp(:,6)-btr1(:,6);Rcheck=timetemp(:,6)-btr2(:,6); LcheckNz=find(Lcheck==0);RcheckNz=find(Rcheck==0); if not isempty(LcheckNz)disp('error in reconstructing timeL');end if not isempty(RcheckNz)disp('error in reconstructing timeR');end timeL=timetemp(:,8);timeR=timetemp(:,8); </pre>

dataadjuster.m

```

C:\$research\Matlab\THESIS\dataadjuster.m Page 1
May 12, 2004 7:11:02 AM

function [dataad, qualinfo]=dataadjuster(data,reorder,fig)

% dataadjuster(data,reorder,fig)
%
% Reorders data, and runs data through findoutliers, and then through gapfiller.
% Assumes data is reordered to the order as given by getshodata (for sigther).
% dataad has an extra column at the end with a 2 for twos , and 1 for gapfiller addition
%
%%%%%
% (c)2003 Stacy J Morris - sjm@alum.mit.edu
%
% if nargin<3
% fig=0;
% else
% nargin<2
% error('please enter both data and reorder');
% end

t= Set up
=====
[datalength,dataamountfull_temp]=size(data);
qualinfo=zeros(1,15);

sigthrm=[3;3;3;3;5;5;3;3;5;5;5;5;5;3;3;3;3];

origdata=data;
clear data;

if dataamountfull_temp>length(reorder)
    dataamountfull_temp
    length(fig_or_order)
    error('problem with fig_or_order');
end

lostcols=0;
for i=1:datamountfull_temp
    if reorder(i)>0
        data(:,reorder(i))=origdata(:,i);
        reord(i)=reorder(i);i=i;
    elseif reorder(i)==0
        lostcols=lostcols+1;
    end
end

origdata=data;
dataamountfull=dataamountfull_temp-lostcols;

time=data(:,dataamountfull);
dataamount=dataamountfull-1;

% Find 2 occurrences
=====
[z2,toss]=find(data(:,1),dataamount)==2;
if not(isempty(z2))
    suspectrows=findunique(z2);
    else
        suspectrows=[];
    end

% Run Analysis
=====
```

```

C:\jml\Research\Matlab\THESIS\dataadjuster.m Page 2
May 12, 2004 7:11:00 AM

[readnew,fake=new,newrownums,gaps]=gapfiller(time,fskedata,2*,0.00668); %to get record of ✓
points where gaps are filled

dataadj(:,1)=realnew(:,1);
dataadj(:,2)=(realnew(:,2)*0.1; %0.1 is the flag for "generated" data during time gaps
dataadjlength,cols)=size(dataadj);

for j=1:datarlength
    %%use find outliers %
    datareplace=findoutliers(data(:,j),time,sightr(j),10+j,2,suspectsrows);

    %%Replace outliers %
    if notisempty(datareplace)
        [dtrl,loss]=size(datareplace);
        for i=1:dtrl
            if realnew(i,j)<0
                if datareplace(i,1)==1 datareplace(i,1)=datalength %if very first or v
                ery last needs replacing, set mean for that data
                data(datareplace(i,1))=mean(data(:,j));
            else
                p=polyfit([time(datareplace(i,1)+1,1):time(datareplace(i,1)+1,1)],(da
                ta(datareplace(i,1)+1,1)-data(datareplace(i,1)+1,1))/time(datareplace(i,1)+1,1));
                data(datareplace(i,1),j)=polyval(p,time(datareplace(i,1),1));
            end
            %find row number once gaps are filled
            [r,toss]=find(newrownums==datareplace(i,1));
            %note that data was replaced
            newrownums=r;
            if r>1 newrownums=[r-1,23,23,19];
            twoflag=(r)num2str(datareplace(i,2));
            dataadj(:,2)=base2dec(twoflag,3);
            if datareplace(i,2) ==1
                quainf0(6)=quainf0(1,6)+1;
            elseif datareplace(i,2) ==2
                quainf0(5)=quainf0(1,5)+1;
            end
        end
    end
    %% Graph if desired %%%%%%%%%%%%%
    if fig0
        for i=0:1:4
            figure(fig+i);clf;
        end
        t=figm([fig fig fig fig fig fig fig1 fig1 fig1 fig1 fig1 fig1 fig1 fig1 fig1]);
        figm([fig fig fig fig fig1 fig1 fig1 fig1 fig1 fig2 fig2 fig2 fig2 fig3 fig3 fig4 fig4]);
        title(m, '1'/'h' ' 2'/'mm' ' 3'/'ml' ' 4'/'heel' ' 5'/' toe
        ' 6'/'ankle' ' 7'/'knee' ' 8'/'gym' ' 9'/'acetab' ' 10'/'gyro x' ' 11'/'acc' ' 12'/'gyro y' ' 13'/'acc' ' 14'/'cap heel' ' 15'/'cap toe' ' 16'/'cap toe' ' 17'/'gyro z' ' 18'/'acc z' ' 20'/' );
    end

```

findoutliers.m

<pre>C:\sjm\Research\Matlab\General\findoutliers.m May 12, 2004 Page 1 7:14:29 AM function datatoreplace=findoutliers(data,time,sigthr,fig,itermax,suspectrows) % findoutliers(data,time,sigthr,fig,itermax) % returns datatoreplace -- indices in first column, and % replaces ones in second column; 1="bad" data; 2="twodata" % % (c)2003 Stacy J Morris - sjm@alum.mit.edu if margin<3 sigthr=3; end if margin>7 sigthr=7; end if margin>10 suspectrows=[]; else if margin>4 sigthr=4; else if margin>2 suspectrows=[]; elseif margin>5 itermax=2; suspectrows=[]; elseif margin>6 suspectrows=[]; end end end if itermax>7 disp('itermax reset to 7'); end if length(data)>length(data); % timetimecondition(time); % get rid of 2's (throw off stddev) for i=1:length(data) data(i,2)=i; % keep original index end dataorig=data; timerorig=time; end [toss]=find(data(:,1)==2); for i=1:length(toss)%chop out the 2! if r2(i)== time(i,:);length(data,:)); data(i,:)=length(data,:)); elseif r2(i)==data.length time(i,:);length(data,:)); data(i,:)=length(data,:)); else r2adj=data.length-length(data); time(i,:);length(data,:)); data(i,:)=length(data,:)); end dataorig(r2(i),1)=1; %for plotting % data(r2(i),1)=0.5*(data(r2(i)+1)+data(r2(i)-1)); end % plot original data if fig>0</pre>	<pre>C:\sjm\Research\Matlab\General\findoutliers.m May 12, 2004 Page 2 7:14:29 AM figure(fig) clf; subplot(2,1,1); hold on; plot(timerorig,dataorig(:,1),'b') plot(timerorig,dataorig(:,1),'b.') end iter=0; repodata=[]; col=['k' '-' g--'r--'; 'g--'r--k' 'r' g; 'r' r']; sigthr=sigthr; srblength=suspectrows; srind=1; while iter<itermax %for each process, find large spikes overall if iter>0 & fig>0 Ddata=difff(data(:,1)); subplot(2,1,2); hold on; plot(time,[Ddata]); end lengthdatastart=length(data); srind=1; for i=1:lengthdatastart-2 srflag=0; iadj=lengthdatastart-length(data(:,i)); %i in adjusted data -- need to repeat the same row, otherwise we will skip evaluating the next point! iactdata=(i-adj),2]; iactnext=data((i-adj)+1,2); if srind<srbl & iactnext==suspectrows(srind) %in case of the two -- need to jump forward rd=1; srind=srind+1; if srind>srbl & iactnext==suspectrows(srind) %if the next row is suspicious, flag it (we are evaluating the next row) srflag=1; srind=srind+1; end imid=75; %for safety, check imid each time if length(data)<2*imid imid=floor(0.5*length(data))-2; end if (i-adj)<imid Ddata=difff(data(:,imid*2,1)); data1_loc=(i-adj)); elseif (i-adj).length(data(:,1))-imid Ddata=difff(data((length(data(:,1))-imid*2:length(data(:,1)),1)); data1_loc=(i-adj)-(length(data(:,1))-imid*2)); else Ddata=difff(data((i-adj)-imid+1:(i-adj)+imid-1,1)); data1_loc=imid; end end if iadj>0 Ddata=difff(data(:,imid*2,1)); data1_loc=(i-adj)); elseif (i-adj).length(data(:,1))-imid Ddata=difff(data((length(data(:,1))-imid*2:length(data(:,1)),1)); data1_loc=(i-adj)-(length(data(:,1))-imid*2)); else Ddata=difff(data((i-adj)-imid+1:(i-adj)+imid-1,1)); data1_loc=imid; end end if fig>0 subplot(2,1,1); plot(timerorig,'rs','cs','gs','rd','cd','gd','ev','cv','gv','r','c','g','r','rc'); dotcol1='g';'k';'m';'k';'g';'d';'gv';'kv';'g';'k';'g';'k';'g';'r';'k'; dotcol1='g';'k';'m';'k';'g';'d';'gv';'kv';'g';'k';'g';'k';'g';'r';'k'; dotcol1ter=21,i,'rs', 'bs'; for i=1:sd21 if sortrepldata21(i,2)==iter+1 replnote=2; else replnote=1; end if i==1 datatoreplace=[datatoreplace;sortrepldata21(i,1),repnote]; if fig>0 subplot(2,1,1); plot(timerorig(sortrepldata21(i,1)),dotcol1(sortrepldata21(i,1))); end elseif sortrepldata21(i,1)==sortrepldata21(i-1,1) datatoreplace=[datatoreplace;sortrepldata21(i,1),repnote]; if fig>0 subplot(2,1,1); plot(timerorig(sortrepldata21(i,1)),dotcol1(sortrepldata21(i,1))); end end end end end</pre>
<pre>C:\sjm\Research\Matlab\General\findoutliers.m May 12, 2004 Page 3 7:14:29 AM if fig>0 time orig i's and timerorig below (just for plotting) if iact1 time_i_loc=1; elseif iact>data.length-10 time_i_loc=data.length-10; else time_i_loc=iact; end end sigDd_it=std(Data_iterr); meanDd_it=mean(Data_iterr); Dd_i_flag=(Data_iterr(Data_i_loc,1)>=sigthr*sigDd_it+meanDd_it Data_iterr (Data_i_loc,1)<=meanDd_it-sigDd_it) & Data_iterr(Data_i_loc,1)>0; Dd_iplus_flag=(Data_iterr(Data_i_loc+1,1)>=sigthr*sigDd_it+meanDd_it Data_iterr (Data_i_loc+1,1)<=meanDd_it-sigDd_it) & Data_iterr(Data_i_loc+1,1)>0; if Dd_i_flag & Dd_iplus_flag & diffsigns(Data_iterr(Data_i_loc,1),Data_iterr(Data_d at_a_i_loc+1,1)) repldata=repldata; data((i-adj),1,2) iter+2]; data=[data((i-adj),1); data((i-adj)+2:length(data,:));] %get rid of this data if fig>0 subplot(2,1,2) plot((time(i-adj)-time(i-loc-10):timeorig(time_i_loc+10), +(sigthr*sigDd_it + er sigthr*sigDd_it)*meanDd_it,col(i-1,1)); plot((timeorig(time_i_loc-10):timeorig(time_i_loc+10), -(sigthr*sigDd_it + er sigthr*sigDd_it)*meanDd_it,col(i-1,1)); else srflag Dd_i_flag=(Data_iterr(Data_i_loc,1)>(sigthr-1)*sigDd_it+meanDd_it Dd_it er(Data_i_loc,1)<=meanDd_it-sigDd_it); Dd_iplus_flag=(Data_iterr(Data_i_loc+1,1)>(sigthr-1)*sigDd_it+meanDd_it er(Data_iterr(Data_i_loc+1,1)<=meanDd_it-sigDd_it); if Dd_i_flag & Dd_iplus_flag & diffsigns(Data_iterr(Data_i_loc,1),Data_iterr(Data_i_loc+1,1)) repldata=repldata; data((i-adj),1,2) iter+2]; data=[data((i-adj),1); data((i-adj)+2:length(data,:));] %get rid of this his data if fig>0 subplot(2,1,2) plot((time(i-adj)-time(i-loc-10):time(i-loc+10), +((sigthr-1)*sigDd_it + er (sigthr-1)*sigDd_it)*meanDd_it,col(i-1,1)); plot((time(i-loc-10):time(i-loc+10), -((sigthr-1)*sigDd_it + er (sigthr-1)*sigDd_it)*meanDd_it,col(i-1,1)); end end end if isempty(repldata)</pre>	<pre>C:\sjm\Research\Matlab\General\findoutliers.m May 12, 2004 Page 4 7:14:29 AM iter=itermax; else iter=iter+1; end end repodata=[repodata;r2_ ones(length(r2),1)*(iter*2+1)]; %add in 2s [sd21,toss]=size(repodata); if sd21>1 sortrepldata2=sortrows(repodata,2); sortrepldata21=sortrows(sortrepldata2,1); else sortrepldata21=repodata; end datatoreplace=[]; % dotcol1='r';'o';'g';'rs';'cs';'gs','rd','cd','gd','ev','cv','gv','r','c','g','r','rc'; % dotcol1='g';'k';'m';'k';'g';'d';'gv';'kv';'g';'k';'g';'k';'g';'r';'k'; dotcol1='g';'k';'m';'k';'g';'d';'gv';'kv';'g';'k';'g';'k';'g';'r';'k'; dotcol1ter=21,i,'rs', 'bs'; for i=1:sd21 if sortrepldata21(i,2)==iter+1 replnote=2; else replnote=1; end if i==1 datatoreplace=[datatoreplace;sortrepldata21(i,1),repnote]; if fig>0 subplot(2,1,1); plot(timerorig(sortrepldata21(i,1)),dotcol1(sortrepldata21(i,1))); end elseif sortrepldata21(i,1)==sortrepldata21(i-1,1) datatoreplace=[datatoreplace;sortrepldata21(i,1),repnote]; if fig>0 subplot(2,1,1); plot(timerorig(sortrepldata21(i,1)),dotcol1(sortrepldata21(i,1))); end end end</pre>

gapfiller.m

<pre>C:\sjm\Research\Matlab\General\gapfiller.m May 12, 2004 function [xnew,ynew,rownums,gaps]=gapfiller(x,y,in,fig) %GAPFILLER(x,y,in,fig,Np,BF,figt) % % x: time, y: data, in: sample interval (default: 2/150) % fig: value for the figure number for graph of ynew, or 0 for no graphs % (or if none of the other variables are used, it can be left blank) % % figt: like fig, but to plot xnew % % (c)2003 Stacy J. Morris - sjm@alum.mit.edu if nargin<4 fig=0; end %Find gaps gaps={}; xchange=difff(x(:,1)); xchange(:,2)=round(xchange(:,1)/in)-1; [rx,rcos]=find(xchange(:,2)>0); gaps=[rx xchange(rx,2)]; [gL,rcos]=size(gaps); for i=1:datalen rownums(i)=i; end %Fill in gaps warning off MATLAB:polyfit:RepeatedPointsOrRescale; warning off MATLAB:polyfit:PolyNotUnique; % checkmean=max(y); % extremaidiff=max(max(y)-checkmean,checkmean-min(y)); if isempty(gaps) xnew=x; xnew(:,1)=0; ynew=y; rownums=rownums_orig; % disp("No gaps!") end %Graph, if desired if fig>0 figure(fig);clf;hold on; plot(x,y,'b.') axval=axis;axis([axval(1:2) 0 1]); plot(xnew,xnew,'color',[0.1 0.8 0.2]); plot(xnew,ynew,'r.'); plot(x,y,'r') if isempty(gaps) xlabel('No Gaps!'); changefont('arial',12,'x'); end end</pre>	<pre>C:\sjm\Research\Matlab\General\gapfiller.m May 12, 2004 else xnew=[igaps(1,1)]; %first block of good data xnew(1,2)=0; %to start second column ynew(1,igaps(1,1),1); rownums=rownums_orig(1:igaps(1,1),1); inewgaps(1)=1; %inew: where new data will start ... for i=1:gl if gaps(i,2)<4 %fill in three or fewer missing packets ppolyfit([xnew((inew-1),1):(gaps(i,1)+1,1)],{ynew((inew-1),1):y(gaps(i,1)+1,1)},1); inew=inew+1; else for j=1:gaps(i,2) xnew(j,1)=(gaps(i,1)+1)*inj; xnew(j,2)=0; %this is to say this data is generated ynew(j,igaps(i,1),1); rownum=(inew,1)=0; inew=inew+1; end gaps(i,3)=1; %to indicate this gap data was replaced end if i<gl xnew=[xnew x(gaps(i,1)+1:gaps(i+1,1)) zeros(gaps(i+1,1)-(gaps(i,1)+1)+1,1)]; iadd=next block of good data (new values already added in) ynew=[ynew y(gaps(i,1)+1:gaps(i+1,1))]; rownums=rownums_orig(gaps(i,1)+1:gaps(i+1,1)); inewlength(inew)+1; else xnew=[xnew x(gaps(i,1)+1:datalen-1) zeros(datalength-(gaps(i,1)+1),1)]; iadd=last block of good data (new values already added in) ynew=[y(gaps(i,1)+1:datalen-1)]; rownums=rownums_orig(gaps(i,1)+1:datalen-1); end end end %Graph, if desired if fig>0 figure(fig);clf;hold on; plot(x,y,'b.') axval=axis;axis([axval(1:2) 0 1]); plot(xnew,xnew,'color',[0.1 0.8 0.2]); plot(xnew,ynew,'r.'); plot(x,y,'r') if isempty(gaps) xlabel('No Gaps!'); changefont('arial',12,'x'); end end</pre>
--	--

plotadjdata.m

<pre>C:\sjm\Research\Matlab\THESIS\plotadjdata.m May 12, 2004 Page 1 7:18:06 AM function plotadjdata(data,time,adjustments,j,fig) % % plotadjdata(data,time,adjustments,j,fig) % takes in column vector of data, column vector of adjustments (as labeled % by dataadjuster.m, *) identifying data vector's location and fig (i.e. % from gcf), and plots adjustments % % (c)2003 Stacy J Morris - sjm@alum.mit.edu figure(fig);hold on; plot(time,data,'*'); %color*,[0.1 0.6 0.1]); %plot line of data in green data=data*time*adjustments; sdata=sortrows(data,3); %sort by adjustments. sdatal=length(sdata); ms_k=0; ms_g=8; ms_g8=8; ms_r10=10; [r0,cosm]=find(sdata(:,3)==0); %find the zeros - these are original, unchanged data r0=length(r0); %zeros will all be in order ... plot(sdata(r0+1:r0+2),sdata(r0+1:r0+2),'k.','linewidth',0.5,'markersize',ms_k); %plot as black dots instead of blue [r1,r0]=find(sdata(:,3)==0); %find the point ones - these are data added by gapfiller if not (isempty(r1)) r0=r0+1; %ones will all be in order ... plot(sdata(r1+1:r1+2),sdata(r1+1:r1+2),'k*', 'markersize',ms_g,'markerfacecolor','g') %co lor,[1 0] plot(sdata(r1+1:r1+2),sdata(r1+1:r1+2),'kd','markersize',ms_g,'markerfacecolor','g') %co lor,[1 1 0] plot(sdata(r1+1:r1+2),sdata(r1+1:r1+2),'k^','markersize',ms_g8,'markerfacecolor','g') %co lor,[1 0 1] plot(sdata(r1+1:r1+2),sdata(r1+1:r1+2),'k*','markersize',ms_g8,'markerfacecolor','g') %co lor,[1 0 0.5] plot(sdata(r1+1:r1+2),sdata(r1+1:r1+2),'k^','markersize',ms_g8,'markerfacecolor','r') %co lor,[1 0 0.5] plot(sdata(r1+1:r1+2),sdata(r1+1:r1+2),'k^','markersize',ms_g8,'markerfacecolor','r') %plot as red pentagons else disp('uh oh') twoflagj end end</pre>	<pre>C:\sjm\Research\Matlab\THESIS\plotadjdata.m May 12, 2004 Page 2 7:18:06 AM plot(sdata2s(1,2),sdata2s(1,1),'k','linewidth',0.5,'markersize',ms_k); %plot as black lines with dots elseif twoflagj=1 %replaced data plot(sdata2s(1,2),sdata2s(1,1),'kd','markersize',ms_g,'markerfacecolor','g') %co lor,[1 1 0] plot(sdata2s(1,2),sdata2s(1,1),'k^','markersize',ms_g8,'markerfacecolor','g') %co lor,[1 0 1] plot(sdata2s(1,2),sdata2s(1,1),'k*','markersize',ms_g8,'markerfacecolor','g') %co lor,[1 0 0.5] plot(sdata2s(1,2),sdata2s(1,1),'k^','markersize',ms_g8,'markerfacecolor','r') %co lor,[1 0 0.5] plot(sdata2s(1,2),sdata2s(1,1),'k^','markersize',ms_g8,'markerfacecolor','r') %plot as red pentagons else disp('uh oh') twoflagj end end</pre>
---	--

gencalibrations.m

<pre>C:\sjm\Research\Matlab\THESIS\gencalibrations.m May 12, 2004 function gencalibrations(datenames) %GENCALIBRATIONS(datenames) % %datenames should be a matrix with rows [subjID trialID shoehid] %(c)2003 Stacy J Morris -sjm@slim.mit.edu disp('disabled');return % load relevant data load c:\sjm\research\data\alldata\allgsv load c:\sjm\research\data\calibFactors\allmeans load c:\sjm\research\data\calibFactors\fasttch load c:\sjm\research\data\calibFactors\bend2deg load c:\sjm\research\data\calibFactors\gyrocalfac gyro_sfs=(gyrocal_st(:,2)*gyrocal_st(:,2)); %col 1 is recorded zo accel_zos=[0.5*(accel_scalefactor_stack1(1,:)*accel_scalefactor_stack1(2,:))' 0.5*(a\x accel_sfs=[(0.5*(accel_scalefactor_stack1(1,:)*accel_scalefactor_stack1(2,:)))/q' 0.5*(a\x accel_scalefactor_stack2(1,:)*accel_scalefactor_stack2(2,:))/q')]; for i=1:2:length(datenames) disp(['gv0' num2str(datenames(i,1)) ' ' num2str(datenames(i,2)) ', row ' num2str(i)]); eval(['dataL = gv0' num2str(datenames(i,1)) ' ' num2str(datenames(i,2)) ';\n']); eval(['dataR = gv0' num2str(datenames(i,1)) ' ' num2str(datenames(i,2)) ';\n']); subjmeans=meanstd(datenames(i,1)-10,:);subjmeans=meanstd(datenames(i,1)-10,:); if datenames(i,1)<21;stackL=1;stackR=2;stackR=1;end dataL=rows(dataL,22);nonzero=find(dataL(:,22)>0); if nonzero;dataL=da\t tall1=nonzero(1,:);dataL(tall1,:)=dataL(tall1,:)/sum(dataL(tall1,:)); dataL=rows(dataL,22);nonzero=find(dataR(:,22)>0); if nonzero;dataR=da\t tall1=nonzero(1,:);dataR(tall1,:)=dataR(tall1,:)/sum(dataR(tall1,:)); end %calibrate fss heelmed_kg_L=feval(F_smallsfss,dataL(:,1)); heelat_kg_L=feval(F_smallsfss,dataL(:,2)); metmed_kg_L=feval(F_bigfss, dataL(:,3)); metlat_kg_L=feval(F_bigfss, dataL(:,4)); heelmed_kg_R=feval(F_smallsfss,dataR(:,1)); heelat_kg_R=feval(F_smallsfss,dataR(:,2)); metmed_kg_R=feval(F_bigfss, dataR(:,3)); metlat_kg_R=feval(F_bigfss, dataR(:,4)); fssum=metmed_kg_L*metlat_kg_L*heelmed_kg_L*heelat_kg_L; fssum=metmed_kg_R*metlat_kg_R*heelmed_kg_R*heelat_kg_R; %center puds caldataL(:,5:6)=(dataL(:,5)-subjmeans(1,5)*dataL(:,6)-subjmeans(1,6)); caldataR(:,5:6)=(dataR(:,5)-subjmeans(1,5)*dataR(:,6)-subjmeans(1,6)); %center bonds caldataL(:,7:8)=(dataL(:,7)-subjmeans(1,7)*data(:,8)-subjmeans(1,8)); caldataR(:,7:8)=(dataR(:,7)-subjmeans(1,7)*data(:,8)-subjmeans(1,8)); </pre>	<pre>C:\sjm\Research\Matlab\THESIS\gencalibrations.m May 12, 2004 Page 2 7:30:31 AM %calibrate gyros, assuming subjmeans=zero offset (gyroz will be replaced later) %x caldataL(:,13)=(dataL(:,13)-subjmeansL(1,13))/gyro_sfs(1,stackL); caldataR(:,13)=(dataR(:,13)-subjmeansR(1,13))/gyro_sfs(1,stackR); %y caldataL(:,11)=(dataL(:,11)-subjmeansL(1,11))/gyro_sfs(2,stackL); caldataR(:,11)=(dataR(:,11)-subjmeansR(1,11))/gyro_sfs(2,stackR); %z caldataL(:, 9)=(dataL(:, 9)-subjmeansL(1, 9))/gyro_sfs(3,stackL); caldataR(:, 9)=(dataR(:, 9)-subjmeansR(1, 9))/gyro_sfs(3,stackR); %calibrate accels %x caldataL(:,14)=(dataL(:,14)-accel_zos(1,stackL))/accel_sfs(1,stackL); caldataR(:,14)=(dataR(:,14)-accel_zos(1,stackR))/accel_sfs(1,stackR); %y caldataL(:,12)=(dataL(:,12)-accel_zos(2,stackL))/accel_sfs(2,stackL); caldataR(:,12)=(dataR(:,12)-accel_zos(2,stackR))/accel_sfs(2,stackR); %z caldataL(:,10)=(dataL(:,10)-accel_zos(3,stackL))/accel_sfs(3,stackL); caldataR(:,10)=(dataR(:,10)-accel_zos(3,stackR))/accel_sfs(3,stackR); %add in fssum, time for saving caldataL(:,20)=fssum; caldataR(:,21)=dataR(:,21); caldataL(:,21)=dataR(:,21); for f=1:14 figure(1); subplot(3,5,f); plot(dataL(:,21),caldataL(:,f)); figure(2); subplot(3,5,f); plot(dataR(:,21),caldataR(:,f)); end eval(['gv0' num2str(datenames(1,1)) ' ' num2str(datenames(1,2)) ' L_cal=caldataL']);% eval(['gv0' num2str(datenames(1,1)) ' ' num2str(datenames(1,2)) ' R_cal=caldataR']);% emssacis1(['gv0' num2str(datenames(1,1)) ' ' num2str(datenames(1,2)) ' Lcal']);% emssacis2(['gv0' num2str(datenames(1,1)) ' ' num2str(datenames(1,2)) ' Rcal']);% save c:\sjm\research\data\calibrateddata *_cal clear caldataL caldataR; end </pre>
--	---

clippedmean.m

<pre>C:\sjm\Research\Matlab\General\clippedmean.m May 12, 2004 Page 1 7:32:32 AM function cm=clippedmean(data) %removes top and bottom 10% dataL=length(data); toremove=floor(0.1*dataL); datasort=sortrows(data,1); %puts data in sequential order, so that the top / bottom can be removed clippeddata=datasort(toremove+1:dataL-toremove); cm=mean(clippeddata);</pre>
--

linear_integrator.m

```
C:\sjm\Research\Matlab\general\linear_integrator.m          Page 1
May 12, 2004          7:32:09 AM
function int_data=linear_integrator(data,time,initialvalue)
% LINEAR_INTEGRATOR(data,time,initialvalue) integrates the data function v. time,
% with the assumption that DeltaT is small enough that the data function is linear
% between each DeltaT. It returns the integrated function.
%
% (c)2003 Stacy J. Morris - sjm@alum.mit.edu

time_end=length(time);

int_data(1,1)=initialvalue;

for i=2:time_end
    m=(data(i)-data(i-1))/(time(i)-time(i-1));
    b=data(i-1)-m*time(i-1);

    int_data(i,1)=(1/2)*m*(time(i)^2 - time(i-1)^2) + b*(time(i)-time(i-1)) + int_data(i-1,1);
end
```

linear_integrator_endadjust.m

```
C:\sjm\Research\Matlab\THESIS\linear_integrator_endadjust.m      Page 1
May 12, 2004          8:40:01 AM
function [int_data,nudge_value]=linear_integrator_endadjust(data,time,initialvalue,endval
ue)
% LINEAR_INTEGRATOR_ENDADJUST(data,time,initialvalue,endvalue,toporbottom) integrates the ✓
data
% function v. time, with the assumption that DeltaT is small enough that the data function ✓
is linear
% between each DeltaT.
%
% This is for use with data where the zero offset is not absolutely known. It integrates ✓
the data as if it were zero offset, and then checks the last value against endvalue, and iterates nudging the data up/✓
down until the resulting endvalue is less than 1% of fullscale LESS THAN the given endvalue.
%
% It returns the integrated function, and the nudge value (subtract the nudge value to a ✓
djust)
%
% (c)2003 Stacy J. Morris - sjm@alum.mit.edu

%initial integration
int_data=linear_integrator(data,time,initialvalue);

%endlimit=.5%fullscale
endlimit=.001*(max(int_data)-min(int_data));
escapehatch50;
data_adj=data;
lastdiff=0; getoutofjailfree0;smallbutpos=[]; usesmallbutpos=0;
firstrime1;nudge_value0; lastdiff0;nudghistory0;0];
while escapehatch50
    %break out when limit
    if abs(endvalue-last(int_data))<endlimit
        if endvalue>last(int_data) | getoutofjailfree
            break;
        else
            if isempty(smallbutpos)
                smallbutpos=[nudge_value abs(endvalue-last(int_data))];
            elseif abs(endvalue-last(int_data))<smallbutpos(1,2)
                smallbutpos=[nudge_value abs(endvalue-last(int_data))];
            end
        end
    end
    %set nudge value (e.g. if endvalue is 0 and last(int_data) is negative, nudge value w
ill be positive)
    %but, make sure it doesn't become unstable
    %if stable, sign should switch, and diff should get smaller
    if (firstrime1 & ( oppsigns((endvalue-last(int_data)),lastdiff) & abs(endvalue-last(in
t_data))<lesser(abs(endvalue-last(int_data)),abs(lastdiff))) )
        nudge_value=nudge_value-abs(last(int_data));
        lastdiff=(endvalue-last(int_data)); nudghistory=[nudge_value;nudghistory1];
    else
        if getoutofjailfree & isempty(smallbutpos) & abs(last(int_data))<10*endlimit; %
        g. within 1%
            disp(['get out of jail free! - at time ' num2str(last(time))]);break;
        end
    end
```

```
C:\sjm\Research\Matlab\THESIS\linear_integrator_endadjust.m      Page 2
May 12, 2004          8:40:01 AM

%see which failed
%if not abs(endvalue-last(int_data)),lastdiff0) & not(abs(endvalue-last(int_
data))=lesser(abs(endvalue-last(int_data)),abs(lastdiff)))
    %same sign & bigger: move in way wrong direction! ... have iterated into a ba
d bad place = make nudge bigger, and allow the getoutofjailfree card
    if escapehatch<20;getoutofjailfree1;usesmallbutpos1;end; nudge_value=10*nud
gehistory1; lastdiff=(endvalue-last(int_data)); nudghistory=[nudge_value;nudghistory1];
else
    elseif not(oppsigns((endvalue-last(int_data)),lastdiff)) %same sign & smaller =>
        main_nudge=-1*a;
        nudge_value=80*nudghistory1; lastdiff=(endvalue-last(int_data)); nudghist
ory=[nudge_value;nudghistory1];
        else %opposite sign, but bigger = overshoot => adjust nudge_value to halfway betw
een last and two prior (or zero if this is second iteration) = mean(nudghistory)
            nudge_value=mean(nudghistory); lastdiff=(endvalue-last(int_data)); nudghis
tory=[nudge_value;nudghistory1];
        end
    end

    if getoutofjailfree & isempty(smallbutpos) & abs(last(int_data))<10*endlimit; %e.g. w
ithin 1% -- here as well as above in case the round which resulted in the getoutofjailfre
e card is under 10*endlimit
        disp(['get out of jail free! - at time ' num2str(last(time))]);break;
    end

    if usesmallbutpos & not isempty(smallbutpos); nudge_value=smallbutpos1; escapehatch ✓
=1; end

    %apply nudge value
    data_adj=data+nudge_value;

    %re-integrate
    int_data=linear_integrator(data_adj,time,initialvalue);

    escapehatch=escapehatch-1; firstrime0;
    if escapehatch==0;disp([' - at time ' num2str(last(time)) ', escape hatch ✓
flipped']);end
```

simpsonsintegrator

```
C:\sjm\Research\Matlab\General\simpsonsintegrator.m      Page 1
May 12, 2004                                         7:43:10 AM
function [yy_si,t_si]=simpsonsintegrator(fnnname,tt_1,tt_2,timediff,yy_init)

if nargin<3
    timediff=.001;yy_init=0;
elseif nargin<4
    yy_init=0;
end

yy_si={};t_si={};
for i=tt_1:timediff:tt_2
    yy_si=[yy_si quad(fnnname,i-timediff,i)+yy_init]; yy_init=last(yy_si);
    t_si=[t_si; i];
end
```

simpsonsintegrator_endadjust.m

```
C:\sjm\Research\Matlab\THESIS\simpsonsintegrator_endadjust.m      Page 1
May 12, 2004                                         7:41:47 AM
function [yy_si,t_si]=simpsonsintegrator_endadjust(tt_1,tt_2,timediff,yy_init,endvalue)
yy_init=origyy_init;
t_si=[tt_1];
yy_si=[yy_init];t_si=[tt_1];
for i=tt_1:timediff:timediff*tt_2
    yy_si=[yy_si quad(@anglepp_fun,i-timediff,i)+yy_init]; yy_init=last(yy_si);
    t_si=[t_si; i];
end

load c:\sjm\Research\Data\calibfactors\anglepp; tt_pp=(tt_1:timediff/5:tt_2);
yy_orig=pval(pp,tt_pp);

% endlimit = .5 fullscale
endlimits=.001*(max(yy_si)-min(yy_si));
escapehatch=0; nudge_value=0;
while escapehatch>0
    %break once within limit
    if abs(endvalue-last(yy_si))<endlimit & endvalue==last(yy_si);break;end

    %set nudge value (e.g. if endvalue is 0 and last(int_data) is negative, nudge value w/ ill be positive)
    nudge_value=nudge_value+(endvalue-last(yy_si));
    yy_nudged=yy_orig+nudge_value; pp=spline([tt_pp],yy_nudged);
    save c:\sjm\Research\Data\calibfactors\anglepp_nudged pp;

    %re-integrate
    yy_init=yy_init; origyy_si=[yy_init];
    for i=tt_1+timediff:timediff*tt_2
        yy_si=[yy_si quad(@anglepp_fun_nudged,i-timediff,i)+yy_init]; yy_init=last(yy_si);
    end

    %don't want to iterate forever!
    escapehatch=escapehatch-1;
    if escapehatch==0; disp([' - - - at time ' num2str(tt_2) ' ', escape hatch flipped']);end
end
```

accel_integrator.m

<pre>C:\sjm\Research\Matlab\Analysis\accel_integrator.m May 12, 2004 function [accinresults_xroom,accinresults_yroom,accinresults_xshoe]=accel_integrator(M easX,MeasY,time,pitch,stack,subj,boundlocs,fig) if nargin<2 fig=0; end % - % - % Constants % - % - % - % - % - % - % - % - % - % - % g=9.81; %m/s2 % - % - % Load calibration info % - % - % - % - % - % - % - % load c:\sjm\Research\Datasets\calibfactors\accelfac eval(['scales=accel_factor_stack' num2str(stack) ';'']); load c:\sjm\Research\Datasets\calibfactors\velmeans eval(['outputatstart=means_Stack' num2str(stack) '(' num2str(subj-10) ',')']); zo = 0.5*(scales(1,:)+scales(2,:)); % zero offset zf2g= 0.5*(scales(1,:)+scales(2,:)/(9.81)); %scale factor to convert to m/s2 % - % - % Angle set up % - % - % - % - % - % - % - % - % theta=pi/180; %change to right hand rule pi radians Mrests=(outputatstart(1,1)-zo)/(f2g(1)); %outputatstart(1,12)-zo(2)/f2g(2)); alpha1(1)=pi-asin(Mrests(1,1)/g); % Y IS BACKWARDS alpha1(2)=pi-asin(Mrests(1,2)/g); % know from geom that y is okay % alpha3=deg(alpha1)*180/pi phis=(theta+alpha1(1)) theta+alpha1(2)); % - % - % Dynamic set up % - % - % - % - % - % - % - % % - % - % - % - % - % - % - % Gxg*sin(phis(:,1)); Gyg*sin(phis(:,1)); Gxg*sin(phis(:,2)); Gyg*sin(phis(:,2)); Mx=(Mx*X-Gx); %dynname=stat My=(My*Y-Gy); % - % - % Determine Ax, Ay % - % - % - % - % - % - % - % % - % - % - % - % - % - % cpxcos(phis(:,1));rpx=sin(phis(:,1)); cpxcos(phis(:,2));rpy=sin(phis(:,2)); Ax=(Mx.*spy*My.*spx)./(cpx.*spy*cpv.*spx); %sign ... Ay=(My.*cpv*My.*cpv)./(spx.*cpv*cpv); %sign ... % - % - % - % - % - % - % boundlocsX=boundlocs; boundlocsY=boundlocs; for i=1:size(boundlocsX,1) Xvel_shoe(boundlocsX(i,1):boundlocsX(i,2),1)=linear_integrator(Mx(boundlocsX(i,1):bo undlocsX(i,2)),time(boundlocsX(i,1):boundlocsX(i,2)),0); %units in m/s Xvel_room(boundlocsX(i,1):boundlocsX(i,2),1)=linear_integrator(Mx(boundlocsX(i,1):bo undlocsX(i,2)),time(boundlocsX(i,1):boundlocsX(i,2)),0); %units in m/s Yvel_room(boundlocsY(i,1):boundlocsY(i,2),1)=linear_integrator(My(boundlocsY(i,1):bo undlocsY(i,2)),time(boundlocsY(i,1):boundlocsY(i,2)),0); %units in m/s end </pre>	<pre>C:\sjm\Research\Matlab\Analysis\accel_integrator.m May 12, 2004 Ydisp_room(boundlocsY(i,1):boundlocsY(i,2),1)=linear_integrator(Yvel_room(boundlocsY(i,1):boundlocsY(i,2)),time(boundlocsY(i,1):boundlocsY(i,2)),0); %units in m end Xvel_shoe(size(Mx,1))=0;Yvel_room(size(Mx,1))=0;Ydisp_room(size(M x,1))=0; Xdisp_shoe=linear_integrator(Xvel_shoe,time,0); %units in m Xdisp_room=linear_integrator(Xvel_room,time,0); %units in m if fig>0 figure(fig);clf; subplot(2,1,1);hold on;plot(time,Ax Mx);title('X accels'); plot(time(boundlocsX(:,1)),Ax(boundlocsX(:,1)), 'gx');plot(time(boundlocsX(:,2)),A x(boundlocsX(:,2)), 'rx'); subplot(2,1,2);hold on;plot(time,Ay My);title('Y accels'); plot(time(boundlocsY(:,1)),Ay(boundlocsY(:,1)), 'g*');plot(time(boundlocsY(:,2)),A y(boundlocsY(:,2)), 'rx'); figure(1);figure(fig);clf; subplot(2,1,1);hold on;plot(time,[Xvel_room Xvel_shoe]);title('X velocities'); subplot(2,1,2);hold on;plot(time,[Yvel_room Yvel_shoe]);title('Y velocities'); figure(1);figure(fig);clf; subplot(2,1,1);hold on;plot(time,[Xdisp_shoe]);title('X disps'); subplot(2,1,2);hold on;plot(time,[Ydisp_room]);title('Y disps'); end accinresults_xroom=[Ax Xvel_room Xdisp_room time]; accinresults_xshoe=[My Xvel_shoe Ydisp_shoe time]; accinresults_yroom=[Ay Yvel_room Ydisp_room time];</pre>
---	---

gyroz_lintint.m

<pre>C:\sjm\Research\Matlab\THESIS\gyroz_lintint.m May 12, 2004 Page 1 8:36:56 AM function [pli,midptbounds,gyrozcal]=gyroz lintint(gyroz_raw,time,stack,hsto,fig,mghgyrotime e) % - % - % - % - % - % Set up scale factor - % - % - % - % - % - % - % % - % - % - % - % % load c:\sjm\matlab transferred\gyrozcal load c:\sjm\Research\Datasets\calibfactors\gyrozcalfac if stack>1 recordedshift=gyrozcal_st1(3,1); scalefactor=gyrozcal_st1(2,2); else recordedshift=gyrozcal_st2(3,1); scalefactor=gyrozcal_st2(2,2); end % - % - % - % - % - % Find zero offset - % - % - % - % % - % - % - % L=length(gyroz_raw);ngyroz=mean(gyroz_raw);nf=0.001; abovemean=0;stophereformean=10; for i=L:-1:1 walk forward until significantly more than mean if gyroz_raw(i)>ngyroz*10*nf abovemean;i;break end end for i=abovemean:-1:greater(10,greater(floor(.75*abovemean),abovemean-30)) walk backward until close to OR first occurrence of: 10 steps back / .75 signmore if gyroz_raw(i)<ngyroz*10*nf stophereformean;i;break end end zoclippedmean=gyroz_raw(1:stophereformean); gyroz=gyroz_raw(1:L); % - % - % - % - % - % Find integration bounds - % - % - % - % % - % - % - % boundsgyrozlinboundfinder(gyroz,time,hsto,nf); [bL,toss]=size(bounds);[htL,toss]=size(hsto); ht=toss; if ht>L-1 bL=L-1; ht=L; bLht=bL;boundsinit=bounds; iter2=inf;nf=[5;2]; while iter2>0 bounds=gyrozlinboundfinder(gyroz,time,hsto,nf*incrnf(iter)); [bt,toss]=size(bounds); if bt==ht;1; break; end iter=iter-1; end bounds %using nf is always more accurate! so replace: for ibinit=2:bLht %first gets tossed [toss,loc]=min(ahs(boundsinit(ibinit,1)-bounds(:,1))); bounds(loc,:)=boundsinit(ibinit,1); end for ibinit=1:bLht-1 %last gets tossed [toss,loc]=min(ahs(boundsinit(ibinit,2)-bounds(:,2))); bounds(loc,2)=boundsinit(ibinit,2); end end </pre>	<pre>C:\sjm\Research\Matlab\THESIS\gyroz_lintint.m May 12, 2004 Page 2 8:36:56 AM % - % - % - % - % % - % - % - % if fig>0 figure(fig);clf; subplot(2,2,1);hold on; plot(midptbounds(1,:)/scalefactor,'b*'); plot(t,y,'k'); plot(time,gyroz/scalefactor,'b.') raxvalgaxis; plot(time(bounds(:,1)),gyroz(bounds(:,1))/scalefactor,'r.'); plot(time(bounds(:,2)),gyroz(bounds(:,2))/scalefactor,'g.'); plot(midptbounds(1,:)/scalefactor,'y*'); plot(midptbounds(1,:)/scalefactor,'y*'); axis([time(1)-1 last(time)+1 axval(3:4)]); subplot(1,2,2);hold on; plot(mghgyrotime(:,2),mghgyrotime(:,1),'r','LineWidth',2); axvalm=axis; plot(pitch_1_cmzo,'k','LineWidth',2); axis([axval(1:2) axval(3:4)]); end pli=[pitch_1_cmzo',time];</pre>
--	---

gyrozlinint_withsplinem.m

```
C:\sjm\Research\Matlab\THESIS\gyrozlinint_withsplinem.m Page 1
May 12, 2004 8:45:22 AM
function [pli,psi,midptbounds,gyrocal,splinefitpts]=gyrozlinint_withsplinem(gyroz_raw,tim
e,stack,hsto,fig,mghyrotzime)
%---^--^--^--^--^--^-- Set up scale factor --^--^--^--^--^--^--^--^--^--^--^
% load c:\jm\matlab\transferred\gyrocalfac
load c:\jm\Research\DATA\calibfactors\gyrocalfac

if stack==1
    recordedshift=gyrocal_st1(3,1);
    scalefactor=gyrocal_st1(3,2);
else
    recordedshift=gyrocal_st2(3,1);
    scalefactor=gyrocal_st2(3,2);
end

%---^--^--^--^--^--^-- Find zero-offset --^--^--^--^--^--^--^--^--^--^
Llength(gyroz_raw);

mgzmean=mean(gyroz_raw);
nf=.001;

abovemean0=stophereformean10;
for i=1:toss forward until significantly more than mean
    if gyroz_raw(i)>mgz+10*nf
        abovemean=i;break
    end
end

% [abovemean floor(.75*abovemean) abovemean-30]
for i=abovemean+1:toss greater(10,greater(10,.75*abovemean),abovemean-30) % walk backward
until next to last occurrence of: 30 steps back /.75 signmore
    if gyroz_raw(i)>mgz+
        stophereformean1;break
    end
end

%=clippedmean(gyroz_raw(listphereformean));

tin case data doesn't start with the foot flat on the floor (most do, so above procedure
works well), use recordedshift instead, and allow a bigger noise floor
if abs(zo-recordedshift)>.05
    zo=recordedshift;nf=.05;
end

gyroz=gyroz_raw-zo;

%---^--^--^--^--^--^-- Find integration bounds --^--^--^--^--^--^--^--^--^--^
bounds=gyrozlinintboundfinder(gyroz,time,hsto,nf);
[ht,toss]=size(bounds);[htL,toss]=size(hsto);

if bl<ht-1
    blinttosh=boundsinit;bounds
    iter2=incrnf(5/2);
    while iter2>
        bounds=gyrozlinintboundfinder(gyroz,time,hsto,nf*incrnf(iter));
        [htL,toss]=size(bounds);
        if bl>htL1;break;end
    end
end

C:\sjm\Research\Matlab\THESIS\gyrozlinint_withsplinem.m Page 2
May 12, 2004 8:45:22 AM
%---^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^
function [pli,psi,midptbounds,gyrocal,pitch_li]=gyrozlinint_withsplinem(gyroz_raw,tim
e,stack,hsto,fig,mghyrotzime)
% Using nf is always more accurate! so replace:
for ibinit=2:htL1 if first gets tossed
    [toss,loc]=min(abs(boundsinit(ibinit,1)-bounds(:,1)));
    bounds(loc,1)=boundsinit(ibinit,1);
end
for ibinit=ibinit-1:last gets tossed
    [toss,loc]=min(abs(boundsinit(ibinit,2)-bounds(:,2)));
    bounds(loc,2)=boundsinit(ibinit,2);
end
end

if isempty(bounds) | bl==2;if fig0;figure(fig);clf; subplot(2,2,1);plot(time,gyroz);psi=[ 
];psi=[];midptbounds=0;gyrozcal=gyroz/scalefactor;end;return;end

for i=1:bl
    midpt=round(0.5*(bounds(i,1)+bounds(i-1,2)));
    midptbounds(i-1,2)=midpt;midptbounds(i,1)=midpt;
end
bounds=bound(2:bl-1,:);
midptbounds=midptbounds(2:bl-1,:);rbL=bl-2;

%---^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^
% Linear Integration --^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^
for i=1:bl
    [pitch_li(midptbounds(i,1):midptbounds(i,2)),nudge(i,1)]=linear_integrator_andadjust(
    gyroz(midptbounds(i,1):midptbounds(i,2),1)/scalefactor,(time(midptbounds(i,1)):midptbounds
    (i,2),1),0,0);
end
pitch_li(1)=pitch_li(midptbounds(1,2));% other midptbounds get set to 0 during follow
pitch_integration

%---^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^
% Set up gyrocal,pitch_li for return argument--^--^--^--^--^--^--^--^--^--^
gyrozcal=gyroz/scalefactor;
pitch_li=pitch_li';

%---^--^--^--^--^--^-- Look for significant gaps, use splines as necessary --^--^
%look up gaps
usesplines=[];
for i=1:toss
    %check data rate - if less than 70 Hz, then check for consecutive dropouts of 4 pts o
    r longer - if any exist, use splines for this step
    if (midptbounds(i,2)-midptbounds(i,1))/(time(midptbounds(i,2))-time(midptbounds(i,1)))<
    >70
        for m=midptbounds(i,1):1:midptbounds(i,2)
            if ((time(m)-time(m-1))>*(2*.00668) %dropout of 4 pts = gap of 5*.0134
                usesplines(i,1)=1;
            end
        end
    end
end

%use splines if necessary
if isempty(usesplines)
    if length(usesplines)>bl; usesplines(bl,1)=0;end %make same length as bl (if not, tha
n latter values should be zero)
    %---^--^--^--^--^--^-- Set up splines --^--^--^--^--^--^--^--^--^--^
C:\sjm\Research\Matlab\THESIS\gyrozlinint_withsplinem.m Page 3
May 12, 2004 8:45:22 AM
%---^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^
tt=[time(1):0.01:last(time)];tt=Llength(tt);
pp=spline(time,gyroz/scalefactor);
ypp=pval(pp,tt);
save c:\jm\Research\DATA\calibfactors\anglapp pp

spline(midptbounds(:,1)=findimponts((time(midptbounds(:,1)),tt));
spline(midptbounds(:,2)=findimponts((time(midptbounds(:,2)),tt));

%---^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^
% Simpson's Integration --^--^--^--^--^--^--^--^--^--^--^--^--^--^--^
pitch_si=pitch_li(1:midptbounds(1,1)-1);time_si=(time(1:midptbounds(1,1))-1);%set init
ial zeros using pitch_li & original time
ttplot();yy2plot();

for i=1:bl
    if usesplines(i)>0 %only integrate using spline if marked, otherwise, store
        impnts=[];impnts(i)=impnts(i);
        for m=midptbounds(i,1):midptbounds(i,2)-1
            if ((time(m)-time(m-1))>*(2*.00668) %dropout of 4 pts = gap of 5*.0134
                impnts=[impnts m];
            end
        end
        endpt=impnts(end);
        [ipl,toss]=size(impnts);
        impnts=[];
        look at impnts: if fewer than 5 points between two impnts, combine
        getnewfrom();
        for p=ipl-1
            if impnts(p+1,1)-impnts(p,2)>5 %time(impnts(p+1,1))-time(impnts(p,2))>10 *
*2*.00668
                impnts(p+1,1)=impnts(p,1); getridofrows=[getridofrows];
            end
        end
        impnts(getridofrows,:)=[]; [ipl,toss]=size(impnts);

        if <5 points after start / before end - then start/end at start/end
        if impnts(1,1)-midptbounds(1,1)<
            impnts(1,1)-midptbounds(1,1);
        end
        if midptbounds(1,2)-impnts(ipl,2)<
            impnts(ipl,2)-midptbounds(1,2); %not -1 because last point is not kept af
ter integration
        end
        impnts(ipl+1,:)=(midptbounds(1,2)-1 midptbounds(1,2)-1); %for last sweep thro
ugh below
        if fig0; spline(impnts(:,1)=findimponts((time(impnts(:,1)),tt); splineintpt
s(:,2)=findimponts((time(impnts(:,2)),tt); end

        pitch_si=[pitch_si;pitch_li(midptbounds(1,1):impnts(1,1))];time_si=[time_si;t
ime(midptbounds(1,1):impnts(1,1))]; Use 11 data
        for p=1:impnts(1,1)
            if time_si(p)<temp, time_si(p)=temp;simponintegrator.withanglapp_andadjust2(ti
me(impnts(p,1)),time(impnts(p,2)),pitch_li(1:impnts(p,1)),pitch_li(1:impnts(p,2)));
            pitch_si=[pitch_si;pitch_si_temp(2:length(pitch_si_temp)-1)];pitch_li(1:p
ts(p,2):impnts(1,1)); %get rid of last, and replace with pitch_li
            time_si=[time_si(1:temp);time_si(1:temp)-1];pitch_si([1:p-1]);
            time_si=[time_si(1:temp);time_si(1:temp)-1];pitch_si([1:p-1]);
            time(imp
ts(p,2):impnts(1,1)); % if last pt is midptbounds(:,2), last set there will be empty]
        end
    end
end

C:\sjm\Research\Matlab\THESIS\gyrozlinint_withsplinem.m Page 4
May 12, 2004 8:45:22 AM
%---^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^
if fig0; tt2plot=[tt2plot;t(splineintpts(p,1):splineintpts(p,2))];yy2plot()
ot=(yy2plot);yy2plot();
else
    pitch_si=[pitch_si;pitch_li(midptbounds(1,1):midptbounds(1,2)-1)];
    time_si=[time_si(1:midptbounds(1,1));time_si(1:midptbounds(1,2)-1)];
end
end

pitch_si=[pitch_si;pitch_li(midptbounds(bl,2):1)];
time_si=[time_si(1:midptbounds(bl,2)):1];
midptbounds(1,3)=usesplines; its save locations where splines were used (note, these
midptbounds won't work for pitch_si-- will have to use times to find correlating points,
but this is easy to do when needed - no need to store)
else
    pitch_si();time_si(); leave empty if not used
    midptbounds(1,3)=0; tset third column of midptbounds to all 0s (no splines used)
end

%---^--^--^--^--^--^-- Plot, if desired --^--^--^--^--^--^--^--^--^--^--^
if fig0
    figure(fig);clf;
    subplot(2,2,1);hold on;
    plot(time,gyroz,'b');
    if not isempty(usesplines);plot(tt2plot,yy2plot,'k');splineintpts=[yy2plot tt2plot];re
    lease;
    else
        pitch_si();
    end
    subplot(2,2,2);hold on;
    if not isempty(usesplines);plot((time_si,pitch_si,'-','LineWidth',2);axval=axis;
    plot((time,gyroz,'b','LineWidth',2);axval=axis;
    plot((time(bound(1,1)),gyroz(bound(1,1)),'-',scalefactor,'r');
    plot((time(bound(1,2)),gyroz(bound(1,2)),'-',scalefactor,'r');
    plot((time(bound(1,3)),gyroz(bound(1,3)),'-',scalefactor,'r');
    plot((time(bound(1,4)),gyroz(bound(1,4)),'-',scalefactor,'r');
    axis((time(1)-1 last(time)+1 axval(3:4));
    stretchgraphs(2,2);

    subplot(1,2,2);hold on;
    plot(mghyrotzime(:,2),mghyrotzime(:,1),'r','LineWidth',2);axval=axis;
    if not isempty(usesplines);plot((time_si,pitch_si,'-','LineWidth',2);axval=axis;
    plot((time,pitch_li,'b','LineWidth',2);axval=axis;
    plot((time_si,pitch_si,'-','LineWidth',2);axval=axis;
    axis(axval(1:2) axval(3:4));
    stretchgraphs(2,2);

end
%---^--^--^--^--^--^-- Set up pitch results for return --^--^--^--^--^--^--^--^--^--^
%%%
pli=[pitch_li,time];psi=[pitch_si,time_si];

```

gyroz_postspline.m

```
C:\sjm\Research\Matlab\THESIS\gyroz_postspline.m          Page 1
May 12, 2004      7:45:25 AM
function [splineweaksH,splineweaksL,biggestgap]=gyroz_postspline(glinint,time,fig)
[peak_resultsH,peak_resultsL,biggestgap]=pitchpeakfinder(glinint,time,1);
peak_results=[peak_resultsH;peak_resultsL];

if notisempty(peak_results)
    peak_results(:,2)=findtimepoints(peak_results(:,2),time);
    for i=1:size(peak_results,1)
        time=peak_results(i,1);time=(peak_results(i,3)+4,1)';%
        gyrospline=eline((time(peak_results(i,3)-4:peak_results(i,3)+4,1),glinint(peak_results(i,3)-4:i*peak_results(i,3)+4,1),time spline);%+4 = space of 8 = just >1 sec total
        if peak_results(i,1)>0 [splineweaks(i,1),newtimept]=max(gyrospline);else [splineweaks(i,1),newtimept]=min(gyrospline);end
        splineweaks(i,2)=timepline(newtimept);

        if fig>0figure(fig);hold on;plot(time,glinint,'color',[.3 .3 .45]);plot(timepline,gyrospline,'g');plot(time,glinint,'b');
        plot(splineweaks(:,2), splineweaks(:,1),'s','markeredgecolor','y','markerface color','r');
        end
    end

    splineweaksH=[splineweaks(1:size(peak_resultsH,1),:)];
    splineweaksL=[splineweaks(size(peak_resultsH,1)+1:size(peak_results,1),:)];
    peak_result<
    sL(:,3));
    else splineweaksH=[];splineweaksL=[];
end
```

hstos_fsrsum_spline.m

```

C:\sjm\Research\Matlab\THEESIS\hstos_fsrsum_spline.m      Page 1
May 12, 2004          8:57:31 AM

function hstosfsrsum_spline(sumfsr,time,yy,tt,fig)
%<<<<<< use original data to find points above "flats" (-approximate no load) <<<<<<
%<<<<<< min(sumfsr),oligopeacean(sumfr));
flats=find(sumfsr>localmax(1));dflats=diff(flat);
nonflats=find(~flat);hsto_sf_vicinity=[flats(nonflats)+1 flats(nonflats)-1];[htvl,t
osz]=size(hsto_sf_vicinity);

hsto_sf_vicinity_check([hsto_sf_vicinity(:,1)-hsto_sf_vicinity(:,1) (diff(hsto_sf_vicinity
    <,1)) diff(hsto_sf_vicinity(:,2))-1000 1000]);
time_for_hsto_sf_vicinity_check([time(hsto_sf_vicinity(:,2))-time(hsto_sf_vicinity(:,1))
    < (diff(time(hsto_sf_vicinity(:,1)),1) diff(time(hsto_sf_vicinity(:,2)),1))]);
htvl=hstovl+htvl;

hsto_sf_vic_adj=hsto_sf_vicinity(htvl,numadj=0);lastadj=-1;
for i=1:htvl
    if sum(hstovl(i,:))<=1 % seconds (under 6 Hz) if no packets dr
        opeed
        i=(i*0.00668 + lastadj)-1; % ( i*htvl + lastadj)-i+1 & ((htso_sf_vicinity(i,2)<12
        & time_for_hsto_sf_vicinity_check(i,2)<20*2*.00668) | (htso_sf_vicinity_check(i,2)<12
        & time_for_hsto_sf_vicinity_check(i,1)<20*2*.00668) )
        if htvl>1
            hsto_sf_vic_adj(i)=htso_sf_vic_adj(i-1-numadj-2,:); hsto_sf_vic_adj(i-numadj-1,
            else
                hsto_sf_vic_adj(i)=htso_sf_vic_adj(i-1-numadj-1,:);hsto_sf_vic_adj(i-numadj,1)
            hsto_sf_vic_adj(i-numadj+1,2);hsto_sf_vic_adj(i-numadj+2:htvl,:); htvl=htvl-1;numadj=
            numadj+1;lastadj=i;
        end
    end
end
%<<<<<< convert time points to spline-domain <<<<<<
for i=1:htvl
    hsto_sf_vicinity(:,i)=findtimepoints(time(hsto_sf_vicinity(:,i)),tt);
end
%<<<<<< check first time point (can be falsely triggered due to weight-shifting) <<<<<<
if tt(hsto_sf_vicinity(2,1))-tt(hsto_sf_vicinity(1,1))> 2*mean(diff(tt(hsto_sf_vicinity(2:htvl,1)),
1))
    hsto_sf_vicinity=hsto_sf_vicinity(2:htvl,:);htvl=htvl-1;
end
%<<<<<< Use Dyy to find heel strike <<<<<<
Dyy=dyy(yt1,0);
for i=1:htvl
    if i==1:hstachk>geteet([hsto_sf_vicinity(1,1)-round(mean(hsto_sf_vicinity(2:htvl,1))-hsto_sf_vicinity(1,1))+100
    (htvl-1,2)));relmh=hstachk-localmax(1)-1+hstachk
    [toss,localmax]=max(Dyy(hstachk));
    for hmall=hmachk:localmax(1)-1:hstachk
        if (Dyy(hmall)<0,.005
            htvl(i,1)=vtt(hmall); hsto(i,3)=hmall;
            break;
        elseif hmall==hstachk
    end
end

```

```

C:\sjm\Research\Matlab\THEESIS\hstos_fsrsum_spline.m      Page 2
May 12, 2004          8:57:31 AM

localhsmall=find(abs(Dyy(hback:hback+localmax(1)-1))==min(abs(Dyy(hback:hback
    < localmax(1)-1)));
hsto(i,1)=htvl(hback+localhsmall(1)-1); hsto(i,3)=hback+localhsmall(1)-1; break
end
end
htvl(i,2)=htvl(1)*time(i);htvl(i,4)=htvl(i,1);
%<<<<<< Use Dyy to estimate toe off <<<<<<
tf_tz_diffs=[];
for i=1:htvl
    tstart=hsto_sf_vicinity(i,1);
    if jnt>htvl(i,1)&tf_tz_diffs(jnt)=0;
        hsto(i,4)=htvl(i,1)+htvl(i,2);
        if abs(Dyy(htvl(i,1))-htvl(i,2))<.005
            break;
        elseif htvl(i,1)-htvl(i,2)>mean(htvl(i,1:-1));
            htvl(i,1)=htvl(i,2);
        end
    end
    localmin=find(Dyy(tstart:tforw)==min(Dyy(tstart:tforw)));
    for tsmall=tstart:last([localmin-1:tforw
        if abs(Dyy(tsmall))<.005
            htvl(i,4)=tsmall;
            break;
        elseif tsmall==htvl(i,2)
            localmin=find(abs(Dyy(tstart+last([localmin-1:tforw]))==min(abs(Dyy(tstart
            +last([localmin-1:tforw];
            hsto(i,1)=htvl(tstart+last([localmin-1:last([localmin-1]);
            hsto(i,4)=tstart+last([localmin-1:last([localmin-1]);
            break;
        end
    end
%<<<<<< Plot if desired <<<<<<
if fig1
    figure(fig1);
    subplot(2,1,1);plot(time,sumfsr,'b');axval=axis;
    plot(-100,-100,'b');hold on; plot(-100,-100,'k');plot(-100,-100,'r');
    plot(-100,-100,'g');plot(-100,-100,'m--');plot(-100,-100,'g--');
    plot(-100,-100,'k');plot(time,sumfsr,'b');
    plot(axval(1,2),[noloadutoff noloadutoff],r:r);
    plot(htso_sf_vicinity(:,1),yhtso_sf_vicinity(:,1),'mo');
    plot(htso_sf_vicinity(:,2),yhtso_sf_vicinity(:,2),'go');
    plot(htso_sf_vicinity(1,1),yhtso_sf_vicinity(1,1),'m');
    plot(htso_sf_vicinity(1,2),yhtso_sf_vicinity(1,2),'g');
    plot(htvl(i,1),htvl(i,3),'m');plot(htvl(i,2),yhtso_sf_vicinity(1,4),'g');
    for htvl=htvl
        plot(htvl(i,1),htvl(i,3),'m--');
        plot(htvl(i,2),htvl(i,2)),axval(3:4),'g--');
        plot(htvl(i,2),htvl(i,4)),axval(3:4),'g--');
    end
end

```

pitch_compare_postspline.m

```

C:\sjm\Research\Matlab\Analysis\pitch_compare_postspline.m    Page 1
May 12, 2004          7:53:00 AM

function pitch_compare_postspline(start, stop, legendon)
load c:\sjm\research\matlab\run02\run02_ssAndB
load c:\sjm\research\matlab\run02\run02_ssCandB

load c:\sjm\research\matlab\run02\run02_c027_coldata_adj
load c:\sjm\research\matlab\run02\run02_coldata_adj
load c:\sjm\research\data\alldata\alldat

r02qifull=r02qi_ss1;r02qi_ss2;r02qi_ss3;r02qi_ss4;r02qi_ss5;

load c:\sjm\research\data\results\r02pitches_ps
if nargin>0;start=1;stop=3;legendon=0;elseif nargin>2;legendon=0;end
if r02pitches_ps>0

for i=start:2:stop length(r02qifull,:)-1;length(r02qifull,:)-50
    for j='gwo';num2str(r02qifull(i,1))<=num2str(r02qifull(i,2)); row = num2str(i);
    ,;

    % Load shoe data
    eval(['data' row]=num2str(r02qifull(i,1))'; num2str(r02qifull(i,2))' 'L_cir1']);
    eval(['shoeL' row]=gwo'; num2str(r02qifull(i,1))'; num2str(r02qifull(i,2))' 'R_cir1']);
    eval(['ph' row]=gwo'; num2str(r02qifull(i,1))'; num2str(r02qifull(i,2))' 'L_cir1']);

    figure(1);
    Left_foot;
    subplot(2,2,1);hold on; if legendon;plot(-10,-10,'b.');//plot(-10,-10,'g.');//plot(-10,-10,'r.');//end...
    [shoeL, shoeR, sbggapR]=gyrozlinint_postspline(dataR(:,15),dataL(:,21),gcf);
    [r1:r2L,r1:r2R]=compressmgh(mghdata(:,5));adj=highlighters(mghdata(r1:r2L,5),mghdata(r1:r2R,5));
    subplot(2,2,3);hold on; if legendon;plot(-10,-10,'color',[.3 .3 .45]);//plot(-10,-10,'b.');//plot(-10,-10,'o.');//end...
    [mghR,r1:r2L,r1:r2R]=mghdata(r1:r2L,3),mghdata(r1:r2R,11);color=[.3 .3 .45];
    subplot(2,2,1);hold on; if legendon;plot(-10,-10,'color',[.3 .3 .45]);//plot(-10,-10,'b.');//plot(-10,-10,'o.');//end...
    [mghL,r1:r2L,r1:r2R]=mghdata(r1:r2L,1),mghdata(r1:r2R,9);color=[.3 .3 .45];
    subplot(2,2,2);hold on; if legendon;plot(-10,-10,'color',[.3 .3 .45]);//plot(-10,-10,'b.');//plot(-10,-10,'o.');//end...
    [mghR,r1:r2L,r1:r2R]=mghdata(r1:r2L,3),mghdata(r1:r2R,11);color=[.3 .3 .45];
    subplot(2,2,3);hold on; if legendon;plot(-10,-10,'color',[.3 .3 .45]);//plot(-10,-10,'b.');//plot(-10,-10,'o.');//end...
    [mghL,r1:r2L,r1:r2R]=mghdata(r1:r2L,1),mghdata(r1:r2R,9);color=[.3 .3 .45];
    plot(mghL,mbggapL)pitchpeakfinder(adjL(:,1),adjL(:,2),2,gcf);
    %gather pitch mins and maxes
    pitchesL=[];
    for p=1:size(mghL,1)
        loc=min(max(phL(p,2)-sheelL(:,2)));
        pitchesL=[p sheelL(loc,:);mbggapL sheelL(loc,:); sbggapL];
    end
    phL=p;
    for p=1:size(mghL,1)
        loc=min(max(phL(p,2)-sheelL(:,2)));
        pitchesL=[p sheelL(loc,:);mbggapL sheelL(loc,:); sbggapL];
        pitches(p)=sheelL(loc,:);
    end
    pitches();
    pitches(1)=sheelL(1,2);
    pitches(2)=sheelL(2,2);
    pitches(3)=sheelL(3,2);
    pitches(4)=sheelL(4,2);
    pitches(5)=sheelL(5,2);
    pitches(6)=sheelL(6,2);
    pitches(7)=sheelL(7,2);
    pitches(8)=sheelL(8,2);
    pitches(9)=sheelL(9,2);
    pitches(10)=sheelL(10,2);
    pitches(11)=sheelL(11,2);
    pitches(12)=sheelL(12,2);
    pitches(13)=sheelL(13,2);
    pitches(14)=sheelL(14,2);
    pitches(15)=sheelL(15,2);
    pitches(16)=sheelL(16,2);
    pitches(17)=sheelL(17,2);
    pitches(18)=sheelL(18,2);
    pitches(19)=sheelL(19,2);
    pitches(20)=sheelL(20,2);
    pitches(21)=sheelL(21,2);
    pitches(22)=sheelL(22,2);
    pitches(23)=sheelL(23,2);
    pitches(24)=sheelL(24,2);
    pitches(25)=sheelL(25,2);
    pitches(26)=sheelL(26,2);
    pitches(27)=sheelL(27,2);
    pitches(28)=sheelL(28,2);
    pitches(29)=sheelL(29,2);
    pitches(30)=sheelL(30,2);
    pitches(31)=sheelL(31,2);
    pitches(32)=sheelL(32,2);
    pitches(33)=sheelL(33,2);
    pitches(34)=sheelL(34,2);
    pitches(35)=sheelL(35,2);
    pitches(36)=sheelL(36,2);
    pitches(37)=sheelL(37,2);
    pitches(38)=sheelL(38,2);
    pitches(39)=sheelL(39,2);
    pitches(40)=sheelL(40,2);
    pitches(41)=sheelL(41,2);
    pitches(42)=sheelL(42,2);
    pitches(43)=sheelL(43,2);
    pitches(44)=sheelL(44,2);
    pitches(45)=sheelL(45,2);
    pitches(46)=sheelL(46,2);
    pitches(47)=sheelL(47,2);
    pitches(48)=sheelL(48,2);
    pitches(49)=sheelL(49,2);
    pitches(50)=sheelL(50,2);
    pitches(51)=sheelL(51,2);
    pitches(52)=sheelL(52,2);
    pitches(53)=sheelL(53,2);
    pitches(54)=sheelL(54,2);
    pitches(55)=sheelL(55,2);
    pitches(56)=sheelL(56,2);
    pitches(57)=sheelL(57,2);
    pitches(58)=sheelL(58,2);
    pitches(59)=sheelL(59,2);
    pitches(60)=sheelL(60,2);
    pitches(61)=sheelL(61,2);
    pitches(62)=sheelL(62,2);
    pitches(63)=sheelL(63,2);
    pitches(64)=sheelL(64,2);
    pitches(65)=sheelL(65,2);
    pitches(66)=sheelL(66,2);
    pitches(67)=sheelL(67,2);
    pitches(68)=sheelL(68,2);
    pitches(69)=sheelL(69,2);
    pitches(70)=sheelL(70,2);
    pitches(71)=sheelL(71,2);
    pitches(72)=sheelL(72,2);
    pitches(73)=sheelL(73,2);
    pitches(74)=sheelL(74,2);
    pitches(75)=sheelL(75,2);
    pitches(76)=sheelL(76,2);
    pitches(77)=sheelL(77,2);
    pitches(78)=sheelL(78,2);
    pitches(79)=sheelL(79,2);
    pitches(80)=sheelL(80,2);
    pitches(81)=sheelL(81,2);
    pitches(82)=sheelL(82,2);
    pitches(83)=sheelL(83,2);
    pitches(84)=sheelL(84,2);
    pitches(85)=sheelL(85,2);
    pitches(86)=sheelL(86,2);
    pitches(87)=sheelL(87,2);
    pitches(88)=sheelL(88,2);
    pitches(89)=sheelL(89,2);
    pitches(90)=sheelL(90,2);
    pitches(91)=sheelL(91,2);
    pitches(92)=sheelL(92,2);
    pitches(93)=sheelL(93,2);
    pitches(94)=sheelL(94,2);
    pitches(95)=sheelL(95,2);
    pitches(96)=sheelL(96,2);
    pitches(97)=sheelL(97,2);
    pitches(98)=sheelL(98,2);
    pitches(99)=sheelL(99,2);
    pitches(100)=sheelL(100,2);
    pitches(101)=sheelL(101,2);
    pitches(102)=sheelL(102,2);
    pitches(103)=sheelL(103,2);
    pitches(104)=sheelL(104,2);
    pitches(105)=sheelL(105,2);
    pitches(106)=sheelL(106,2);
    pitches(107)=sheelL(107,2);
    pitches(108)=sheelL(108,2);
    pitches(109)=sheelL(109,2);
    pitches(110)=sheelL(110,2);
    pitches(111)=sheelL(111,2);
    pitches(112)=sheelL(112,2);
    pitches(113)=sheelL(113,2);
    pitches(114)=sheelL(114,2);
    pitches(115)=sheelL(115,2);
    pitches(116)=sheelL(116,2);
    pitches(117)=sheelL(117,2);
    pitches(118)=sheelL(118,2);
    pitches(119)=sheelL(119,2);
    pitches(120)=sheelL(120,2);
    pitches(121)=sheelL(121,2);
    pitches(122)=sheelL(122,2);
    pitches(123)=sheelL(123,2);
    pitches(124)=sheelL(124,2);
    pitches(125)=sheelL(125,2);
    pitches(126)=sheelL(126,2);
    pitches(127)=sheelL(127,2);
    pitches(128)=sheelL(128,2);
    pitches(129)=sheelL(129,2);
    pitches(130)=sheelL(130,2);
    pitches(131)=sheelL(131,2);
    pitches(132)=sheelL(132,2);
    pitches(133)=sheelL(133,2);
    pitches(134)=sheelL(134,2);
    pitches(135)=sheelL(135,2);
    pitches(136)=sheelL(136,2);
    pitches(137)=sheelL(137,2);
    pitches(138)=sheelL(138,2);
    pitches(139)=sheelL(139,2);
    pitches(140)=sheelL(140,2);
    pitches(141)=sheelL(141,2);
    pitches(142)=sheelL(142,2);
    pitches(143)=sheelL(143,2);
    pitches(144)=sheelL(144,2);
    pitches(145)=sheelL(145,2);
    pitches(146)=sheelL(146,2);
    pitches(147)=sheelL(147,2);
    pitches(148)=sheelL(148,2);
    pitches(149)=sheelL(149,2);
    pitches(150)=sheelL(150,2);
    pitches(151)=sheelL(151,2);
    pitches(152)=sheelL(152,2);
    pitches(153)=sheelL(153,2);
    pitches(154)=sheelL(154,2);
    pitches(155)=sheelL(155,2);
    pitches(156)=sheelL(156,2);
    pitches(157)=sheelL(157,2);
    pitches(158)=sheelL(158,2);
    pitches(159)=sheelL(159,2);
    pitches(160)=sheelL(160,2);
    pitches(161)=sheelL(161,2);
    pitches(162)=sheelL(162,2);
    pitches(163)=sheelL(163,2);
    pitches(164)=sheelL(164,2);
    pitches(165)=sheelL(165,2);
    pitches(166)=sheelL(166,2);
    pitches(167)=sheelL(167,2);
    pitches(168)=sheelL(168,2);
    pitches(169)=sheelL(169,2);
    pitches(170)=sheelL(170,2);
    pitches(171)=sheelL(171,2);
    pitches(172)=sheelL(172,2);
    pitches(173)=sheelL(173,2);
    pitches(174)=sheelL(174,2);
    pitches(175)=sheelL(175,2);
    pitches(176)=sheelL(176,2);
    pitches(177)=sheelL(177,2);
    pitches(178)=sheelL(178,2);
    pitches(179)=sheelL(179,2);
    pitches(180)=sheelL(180,2);
    pitches(181)=sheelL(181,2);
    pitches(182)=sheelL(182,2);
    pitches(183)=sheelL(183,2);
    pitches(184)=sheelL(184,2);
    pitches(185)=sheelL(185,2);
    pitches(186)=sheelL(186,2);
    pitches(187)=sheelL(187,2);
    pitches(188)=sheelL(188,2);
    pitches(189)=sheelL(189,2);
    pitches(190)=sheelL(190,2);
    pitches(191)=sheelL(191,2);
    pitches(192)=sheelL(192,2);
    pitches(193)=sheelL(193,2);
    pitches(194)=sheelL(194,2);
    pitches(195)=sheelL(195,2);
    pitches(196)=sheelL(196,2);
    pitches(197)=sheelL(197,2);
    pitches(198)=sheelL(198,2);
    pitches(199)=sheelL(199,2);
    pitches(200)=sheelL(200,2);
    pitches(201)=sheelL(201,2);
    pitches(202)=sheelL(202,2);
    pitches(203)=sheelL(203,2);
    pitches(204)=sheelL(204,2);
    pitches(205)=sheelL(205,2);
    pitches(206)=sheelL(206,2);
    pitches(207)=sheelL(207,2);
    pitches(208)=sheelL(208,2);
    pitches(209)=sheelL(209,2);
    pitches(210)=sheelL(210,2);
    pitches(211)=sheelL(211,2);
    pitches(212)=sheelL(212,2);
    pitches(213)=sheelL(213,2);
    pitches(214)=sheelL(214,2);
    pitches(215)=sheelL(215,2);
    pitches(216)=sheelL(216,2);
    pitches(217)=sheelL(217,2);
    pitches(218)=sheelL(218,2);
    pitches(219)=sheelL(219,2);
    pitches(220)=sheelL(220,2);
    pitches(221)=sheelL(221,2);
    pitches(222)=sheelL(222,2);
    pitches(223)=sheelL(223,2);
    pitches(224)=sheelL(224,2);
    pitches(225)=sheelL(225,2);
    pitches(226)=sheelL(226,2);
    pitches(227)=sheelL(227,2);
    pitches(228)=sheelL(228,2);
    pitches(229)=sheelL(229,2);
    pitches(230)=sheelL(230,2);
    pitches(231)=sheelL(231,2);
    pitches(232)=sheelL(232,2);
    pitches(233)=sheelL(233,2);
    pitches(234)=sheelL(234,2);
    pitches(235)=sheelL(235,2);
    pitches(236)=sheelL(236,2);
    pitches(237)=sheelL(237,2);
    pitches(238)=sheelL(238,2);
    pitches(239)=sheelL(239,2);
    pitches(240)=sheelL(240,2);
    pitches(241)=sheelL(241,2);
    pitches(242)=sheelL(242,2);
    pitches(243)=sheelL(243,2);
    pitches(244)=sheelL(244,2);
    pitches(245)=sheelL(245,2);
    pitches(246)=sheelL(246,2);
    pitches(247)=sheelL(247,2);
    pitches(248)=sheelL(248,2);
    pitches(249)=sheelL(249,2);
    pitches(250)=sheelL(250,2);
    pitches(251)=sheelL(251,2);
    pitches(252)=sheelL(252,2);
    pitches(253)=sheelL(253,2);
    pitches(254)=sheelL(254,2);
    pitches(255)=sheelL(255,2);
    pitches(256)=sheelL(256,2);
    pitches(257)=sheelL(257,2);
    pitches(258)=sheelL(258,2);
    pitches(259)=sheelL(259,2);
    pitches(260)=sheelL(260,2);
    pitches(261)=sheelL(261,2);
    pitches(262)=sheelL(262,2);
    pitches(263)=sheelL(263,2);
    pitches(264)=sheelL(264,2);
    pitches(265)=sheelL(265,2);
    pitches(266)=sheelL(266,2);
    pitches(267)=sheelL(267,2);
    pitches(268)=sheelL(268,2);
    pitches(269)=sheelL(269,2);
    pitches(270)=sheelL(270,2);
    pitches(271)=sheelL(271,2);
    pitches(272)=sheelL(272,2);
    pitches(273)=sheelL(273,2);
    pitches(274)=sheelL(274,2);
    pitches(275)=sheelL(275,2);
    pitches(276)=sheelL(276,2);
    pitches(277)=sheelL(277,2);
    pitches(278)=sheelL(278,2);
    pitches(279)=sheelL(279,2);
    pitches(280)=sheelL(280,2);
    pitches(281)=sheelL(281,2);
    pitches(282)=sheelL(282,2);
    pitches(283)=sheelL(283,2);
    pitches(284)=sheelL(284,2);
    pitches(285)=sheelL(285,2);
    pitches(286)=sheelL(286,2);
    pitches(287)=sheelL(287,2);
    pitches(288)=sheelL(288,2);
    pitches(289)=sheelL(289,2);
    pitches(290)=sheelL(290,2);
    pitches(291)=sheelL(291,2);
    pitches(292)=sheelL(292,2);
    pitches(293)=sheelL(293,2);
    pitches(294)=sheelL(294,2);
    pitches(295)=sheelL(295,2);
    pitches(296)=sheelL(296,2);
    pitches(297)=sheelL(297,2);
    pitches(298)=sheelL(298,2);
    pitches(299)=sheelL(299,2);
    pitches(300)=sheelL(300,2);
    pitches(301)=sheelL(301,2);
    pitches(302)=sheelL(302,2);
    pitches(303)=sheelL(303,2);
    pitches(304)=sheelL(304,2);
    pitches(305)=sheelL(305,2);
    pitches(306)=sheelL(306,2);
    pitches(307)=sheelL(307,2);
    pitches(308)=sheelL(308,2);
    pitches(309)=sheelL(309,2);
    pitches(310)=sheelL(310,2);
    pitches(311)=sheelL(311,2);
    pitches(312)=sheelL(312,2);
    pitches(313)=sheelL(313,2);
    pitches(314)=sheelL(314,2);
    pitches(315)=sheelL(315,2);
    pitches(316)=sheelL(316,2);
    pitches(317)=sheelL(317,2);
    pitches(318)=sheelL(318,2);
    pitches(319)=sheelL(319,2);
    pitches(320)=sheelL(320,2);
    pitches(321)=sheelL(321,2);
    pitches(322)=sheelL(322,2);
    pitches(323)=sheelL(323,2);
    pitches(324)=sheelL(324,2);
    pitches(325)=sheelL(325,2);
    pitches(326)=sheelL(326,2);
    pitches(327)=sheelL(327,2);
    pitches(328)=sheelL(328,2);
    pitches(329)=sheelL(329,2);
    pitches(330)=sheelL(330,2);
    pitches(331)=sheelL(331,2);
    pitches(332)=sheelL(332,2);
    pitches(333)=sheelL(333,2);
    pitches(334)=sheelL(334,2);
    pitches(335)=sheelL(335,2);
    pitches(336)=sheelL(336,2);
    pitches(337)=sheelL(337,2);
    pitches(338)=sheelL(338,2);
    pitches(339)=sheelL(339,2);
    pitches(340)=sheelL(340,2);
    pitches(341)=sheelL(341,2);
    pitches(342)=sheelL(342,2);
    pitches(343)=sheelL(343,2);
    pitches(344)=sheelL(344,2);
    pitches(345)=sheelL(345,2);
    pitches(346)=sheelL(346,2);
    pitches(347)=sheelL(347,2);
    pitches(348)=sheelL(348,2);
    pitches(349)=sheelL(349,2);
    pitches(350)=sheelL(350,2);
    pitches(351)=sheelL(351,2);
    pitches(352)=sheelL(352,2);
    pitches(353)=sheelL(353,2);
    pitches(354)=sheelL(354,2);
    pitches(355)=sheelL(355,2);
    pitches(356)=sheelL(356,2);
    pitches(357)=sheelL(357,2);
    pitches(358)=sheelL(358,2);
    pitches(359)=sheelL(359,2);
    pitches(360)=sheelL(360,2);
    pitches(361)=sheelL(361,2);
    pitches(362)=sheelL(362,2);
    pitches(363)=sheelL(363,2);
    pitches(364)=sheelL(364,2);
    pitches(365)=sheelL(365,2);
    pitches(366)=sheelL(366,2);
    pitches(367)=sheelL(367,2);
    pitches(368)=sheelL(368,2);
    pitches(369)=sheelL(369,2);
    pitches(370)=sheelL(370,2);
    pitches(371)=sheelL(371,2);
    pitches(372)=sheelL(372,2);
    pitches(373)=sheelL(373,2);
    pitches(374)=sheelL(374,2);
    pitches(375)=sheelL(375,2);
    pitches(376)=sheelL(376,2);
    pitches(377)=sheelL(377,2);
    pitches(378)=sheelL(378,2);
    pitches(379)=sheelL(379,2);
    pitches(380)=sheelL(380,2);
    pitches(381)=sheelL(381,2);
    pitches(382)=sheelL(382,2);
    pitches(383)=sheelL(383,2);
    pitches(384)=sheelL(384,2);
    pitches(385)=sheelL(385,2);
    pitches(386)=sheelL(386,2);
    pitches(387)=sheelL(387,2);
    pitches(388)=sheelL(388,2);
    pitches(389)=sheelL(389,2);
    pitches(390)=sheelL(390,2);
    pitches(391)=sheelL(391,2);
    pitches(392)=sheelL(392,2);
    pitches(393)=sheelL(393,2);
    pitches(394)=sheelL(394,2);
    pitches(395)=sheelL(395,2);
    pitches(396)=sheelL(396,2);
    pitches(397)=sheelL(397,2);
    pitches(398)=sheelL(398,2);
    pitches(399)=sheelL(399,2);
    pitches(400)=sheelL(400,2);
    pitches(401)=sheelL(401,2);
    pitches(402)=sheelL(402,2);
    pitches(403)=sheelL(403,2);
    pitches(404)=sheelL(404,2);
    pitches(405)=sheelL(405,2);
    pitches(406)=sheelL(406,2);
    pitches(407)=sheelL(407,2);
    pitches(408)=sheelL(408,2);
    pitches(409)=sheelL(409,2);
    pitches(410)=sheelL(410,2);
    pitches(411)=sheelL(411,2);
    pitches(412)=sheelL(412,2);
    pitches(413)=sheelL(413,2);
    pitches(414)=sheelL(414,2);
    pitches(415)=sheelL(415,2);
    pitches(416)=sheelL(416,2);
    pitches(417)=sheelL(417,2);
    pitches(418)=sheelL(418,2);
    pitches(419)=sheelL(419,2);
    pitches(420)=sheelL(420,2);
    pitches(421)=sheelL(421,2);
    pitches(422)=sheelL(422,2);
    pitches(423)=sheelL(423,2);
    pitches(424)=sheelL(424,2);
    pitches(425)=sheelL(425,2);
    pitches(426)=sheelL(426,2);
    pitches(427)=sheelL(427,2);
    pitches(428)=sheelL(428,2);
    pitches(429)=sheelL(429,2);
    pitches(430)=sheelL(430,2);
    pitches(431)=sheelL(431,2);
    pitches(432)=sheelL(432,2);
    pitches(433)=sheelL(433,2);
    pitches(434)=sheelL(434,2);
    pitches(435)=sheelL(435,2);
    pitches(436)=sheelL(436,2);
    pitches(437)=sheelL(437,2);
    pitches(438)=sheelL(438,2);
    pitches(439)=sheelL(439,2);
    pitches(440)=sheelL(440,2);
    pitches(441)=sheelL(441,2);
    pitches(442)=sheelL(442,2);
    pitches(443)=sheelL(443,2);
    pitches(444)=sheelL(444,2);
    pitches(445)=sheelL(445,2);
    pitches(446)=sheelL(446,2);
    pitches(447)=sheelL(447,2);
    pitches(448)=sheelL(448,2);
    pitches(449)=sheelL(449,2);
    pitches(450)=sheelL(450,2);
    pitches(451)=sheelL(451,2);
    pitches(452)=sheelL(452,2);
    pitches(453)=sheelL(453,2);
    pitches(454)=sheelL(454,2);
    pitches(455)=sheelL(455,2);
    pitches(456)=sheelL(456,2);
    pitches(457)=sheelL(457,2);
    pitches(458)=sheelL(458,2);
    pitches(459)=sheelL(459,2);
    pitches(460)=sheelL(460,2);
    pitches(461)=sheelL(461,2);
    pitches(462)=sheelL(462,2);
    pitches(463)=sheelL(463,2);
    pitches(464)=sheelL(464,2);
    pitches(465)=sheelL(465,2);
    pitches(466)=sheelL(466,2);
    pitches(467)=sheelL(467,2);
    pitches(468)=sheelL(468,2);
    pitches(469)=sheelL(469,2);
    pitches(470)=sheelL(470,2);
    pitches(471)=sheelL(471,2);
    pitches(472)=sheelL(472,2);
    pitches(473)=sheelL(473,2);
    pitches(474)=sheelL(474,2);
    pitches(475)=sheelL(475,2);
    pitches(476)=sheelL(476,2);
    pitches(477)=sheelL(477,2);
    pitches(478)=sheelL(478,2);
    pitches(479)=sheelL(479,2);
    pitches
```

pitchpeakfinder.m

```

C:\ejm\Research\Matlab\Analysis\pitchpeakfinder.m Page 1
May 12, 2004 7:57:11 AM

function [highpeaks, lowpeaks, biggestgap]=pitchpeakfinder(data,time,mghrshoe,fig)
%highpeaks, lowpeaks = pitchpeakfinder(data,time,mghrshoe,fig)
% mghrshoe=2 =>bel OR mghrshoe=1 =>no
% enter 0 for fig if you do not wish to have graphs
%
% (c)2004 Stacy J Morris

if nargin<4|fig==0||~isequal(nargin,3)|error('not enough arguments!');end
[row,cols]=size(time);
pave0=pup*20*pdwn*5;
if mghrshoe==2
    time=1/(12*ptspace)*8; %/-8 -->-.1
elseif mghrshoe==1
    time=1/(12*ptspace)*4; %/-4 -->-.1
end
if 1==row
    %block at all values of input data with respect to previous
    %value determined "high average" and "low average" (input
    %if data(i) < pdwn
        %vector ftr_o -- see FSinitialization). it uses the following
    %notation:
        %highrow(i)=1; %i below the "low average"
        %elseif data(i) < pup; %0: between the two averages
        %highrow(i)=+1; %i above the "high average"
    end
    end
    %note that only the -1 values are currently used. the oth
    %er are notated for future use.
    highrow(pptsize)=5;
    for i=1:pptsize-1
        highrow(i)=highrow(i)+1;
    end
    lowrow(pptsize)=5;
    for i=1:pptsize-1
        lowrow(i)=highrow(i)-1;
    end
    if notiszero(lowrow)
        lowpts=(lowrow(1):0):r1;
        for i=2:length(lows)
            if lowpts(i,r1)==0
                lowpts(i,r1)=lowrow(i);
            else
                if lowpts(i,r1)>0
                    if lowrow(i)<lowrow(i-1)+gapsize;
                        lowpts(r2,:)=lowrow(i-1);
                        lowpts(r1,1)=lowrow(i);r=r1+1;
                    end
                end
            end
        end
        if lowpts(r2)==0;lowpts(r2,:)=last(lows);end;
        for i=r1:r2
            [localpeak localalloc]=min(data([lowpts(i,1):lowpts(i,2),1]));
            trueloc=lowpts(i,1)+loc
alloc-1;
            if trueloc-ptspace>1||ptspace>trueloc-1||ptspace<head-17-trueloc...
            -2*ptspace;
                elseif trueloc>ptspace*length(data)/ptspacehead+length(data)-trueloc*ptspace
back+trueloc*(length(data)+16)...;
                elseif ptspacehead>ptspace;ptspaceback=ptspace;end
            else
                extragaps=round((time(trueloc)-ptspacehead)/(time(trueloc-ptspaceback)))/(timediff)
                -2*ptspace;
                localpeak localalloc=min(data([lowpts(i,1):lowpts(i,2),1]));trueloc=lowpts(i,1)+loc
alloc-1;
                lowpeaks(i,1)=[localpeak timetime(trueloc) trueloc extragaps];
            end
        end
    end
end

```

```

C:\jian\Research\Matlab\Analysis\pitchpeakfinder.m Page 2
May 12, 2004 7:57:11 AM

if lowpeaks(r,3)>length(data)-2*ptspace+1;lowpeaks(r,:)=[];end; %get rid of pts at start or end (not nec. true max/min)
if not isempty(lowpeaks);if lowpeaks(1,1)<2*ptspace;lowpeaks(1,:)=[];end;end;%get rid of pts at start or end (not nec. true max/min)
if not isempty([lowpeaks]);lowpeaks(:,3)=[];end
else
    lowpeaks=[];%
end
%-----%
%-----%
%-----%
%-----%
%-----%
highs=ceil((highlowr==1));
if not isempty(highs);
    highs(1,1)=highs(1,1)*0;r=r1;
    for i=2:length(highs)
        if highpts(r,i)==0;
            highpts(r,i)=highs(i);
        else
            if i>1;highs(i,1)=highs(i-1)+gapsize;
            highpts(r,2)=highs(i-1);
            highpts(r+1,1)=highs(i);r=r+1;
        end
    end
end
if highpts(r,2)>0;highpts(r,2)=last(highs);end;
for i=1:r
    if localcomps(i,1)>0;
        if trueloc-pptspace<1;pptspaceback=trueloc-lptspacehead+1;trueloc...
localcomps(i,1)=max(data([highpts(i,1):highpts(i,2),1]));trueloc=hightps(i,1)+...
        if trueloc-pptspace<1;pptspaceback=trueloc-lptspacehead+1;trueloc-pspaceback...
        else trueloc=pptspaceback+pptspaceback-pspaceback;end
        back=trueloc-(length(data)-16)...%length(data)-16
        extrapgap=round((time(trueloc)-pptspaceback)+(time(trueloc)-pptspaceback)/(timediff...
->2*ptspace));
        highpeaks(i,:)=localcomps time(trueloc) trueloc extrapgap;
    end
    if highpeaks(i,3)>length(data)-2*ptspace+1;highpeaks(r,:)=[];end;%get rid of pts at start or end (not nec. true max/min)
    if not isempty(highpeaks);if highpeaks(1,3)<2*ptspace;highpeaks(1,:)=[];end;end;%get rid of pts at start or end (not nec. true max/min)
    if not isempty(highpeaks);highpeaks(:,3)=[];end
else
    highpeaks=[];%
end
%-----%
%-----%
%-----%
%-----%
%-----%
biggestgap=max(diff(time));
%-----%
%-----%
%-----%
%-----%
%-----%
if fig;
    figure(fig); hold on;
    plot(time, data,'b.');
    if not isempty(lowpeaks);plot(lowpeaks(:,2), lowpeaks(:,1), 'o', 'markeredgecolor','y',...
    'markerfacecolor','r');end
    if not isempty(highpeaks);plot(highpeaks(:,2), highpeaks(:,1), 'o', 'markeredgecolor','y',...
    'markerfacecolor','r');end
end

```

getmghdataorder.m

```

C:\ejm\Research\Matlab\THESIS\getmhdataorder.m          Page 1
May 12, 2004                                         8:03:06 AM

function mhdata_return=getmhdataorder(vers)

%GETMHDATA(VERS)
%
% Version 1, data returned in this order:
% LeftFoot(x,y,z,roll,pitch,yaw), RightFoot(x,y,z,roll,pitch,yaw),
% LeftShank(x,y,z,roll,pitch,yaw), RightShank(x,y,z,roll,pitch,yaw),
% LeftAnkle(x,y,z), RightAnkle(x,y,z), Time
%
% (c) 2003 Stacy J Morris - sjm@alum.mit.edu

mhdata=readmhdf1;

allargswhos;
[sizeargs,col]=size(allargs);

if sizeargs<2
    error('You must enter a version number and provide data. See help.');
end

[length,toss]=size(mhdata);

if toss>=120
    error('Data is wrong size!')
end

%Assign data names!!!!!!!
% **** read file 6dof_r1.txt for matlab matrix notation (i.e starts at 1 not 0). ****
% ****
%Six DOF Segment Positions
%#1 Left Foot X
%#1 Left Foot Y
%#1 Left Foot Z
footlpos=mhdata(:,1:3);

%#9 Left Shank X
%#10 Left Shank Y
%#11 Left Shank Z
shanklpos=mhdata(:,10:12);

%#54 Right Foot X
%#55 Right Foot Y
%#56 Right Foot Z
footRpos=mhdata(:,55:57);

%#63 Right Shank X
%#64 Right Shank Y
%#65 Right Shank Z
shankRpos=mhdata(:,64:66);

%Six DOF Segment Rotations
%#3 Left Foot Roll
%#4 Left Foot Pitch
%#5 Left FootYaw
footLrot=mhdata(:,4:6);


```

```

C:\smj\Research\Matlab\THEYSIS\getmghdataorder.m Page 2
May 12, 2004 8:03:06 AM

%1 Left Shank Roll
%13 Left Shank Pitch
%14 Left ShankYaw
shankLrot=mghdata(:,13:15);

%57 Right Foot Roll
%58 Right Foot Pitch
%59 Right Foot Yaw

footRrot=mghdata(:,58:60);
%65 Right Shank Roll
%67 Right Shank Pitch
%68 Right Shank Yaw
shankRrot=mghdata(:,67:69);

%Six DOP Joint Positions
%6 Left Ankle X
%7 Left Ankle Y
%8 Left Ankle Z
ankleLpos=mghdata(:,7:9);

%60 Right Ankle X
%61 Right Ankle Y
%62 Right Ankle Z
ankleRpos=mghdata(:,61:63);

%Assign timepoint
time(0)=0;
for i=1:length
    time(i)=i/152-1/152;
end

%Returned Data
mghdata_return=[mghdata(:,1:13) mghdata(:,4:6) mghdata(:,55:57) mghdata(:,58:60) mghdata(:,10:12) mghdata(:,13:15) mghdata(:,64:66) mghdata(:,67:69) mghdata(:,7:9) mghdata(:,61:63)];

```

findmghoutliers.m

```
C:\sjm\Research\Matlab\General\findmghoutliers.m
May 12, 2004                                         Page 1
7:59:06 AM

function datatime=findmghoutliers(data,time,diffthr,fig)
% findmghoutliers(data,time,sightrs,fig,itermax)
% returns [data time] with outliers removed
%
% (c)2003 Stacy J Morris - sjm@alum.mit.edu

if nargin<3|diffthr==2|fig==0;
elseif nargin<4|fig==0|end

% plot original data & diff
Ddata=diff(data);
if fig>0
    figure(fig)
    clf; subplot(2,1,1);hold on;
    plot(time,data,'g');
    subplot(2,1,2);hold on;
    plot(time,[Ddata;r0]);
end

badind=[];
for i=1:length(data)-1
    if abs(Ddata(i))>diffthr
        badind=[badind;i];
    end
end

if not isempty(badind)
    gapsize=0.05 sec
    if length(badind)>1
        removepts=[badind(1) 0];r=1;if badind(2)<badind(1)+gapsize;i=start=2;else;removepts <
        s(1,2)=badind(1);removepts(2,2)=r;start=2;end;
        for i=r+1:length(badind)
            if removepts(r,1)==0;
                removepts(r,1)=badind(i);
            else;
                if badind(i)>badind(i-1)+gapsize;
                    removepts(r,2)=badind(i-1);
                    removepts(r+1,1)=badind(i);r=r+1;
                end
            end
        end
        if removepts(r,2)==0;removepts(r,2)=last(badind);end;
    else
        removepts=[badind badind];r=1;
    end
    for i=r+1:1
        data(removepts(i,1):removepts(i,2),:)=[];time(removepts(i,1):removepts(i,2),:)=[];
    end
end
% plot final data & diff
if fig>0
    subplot(2,1,1);
    plot(time,data,'b');
end
datatime=[data time];

```

genleaveoutids.m

```
C:\sjm\Research\Matlab\THEESIS\genleaveoutids.m
May 12, 2004                                         Page 1
8:14:32 AM

function makehypCmats
disp('disabled')return

load c:\sjm\research\matlab\run02\r02pr04_females;

subgsit=[r02pr04F(:,1) r02pr04F(:,3)];
leavetriplets=ceil((length(r02pr04F(:,1)),9));
for f=1:length(subgsit)
    gnames={'free','dist','pace'};
    for g=1:10:30
        rows=findwools3(subgsit,{femalesnum(f,:) g});
        subg=leavethis(f,g/10)*length(rows);
        eval(['subj_-' gnames(g/10,:) '-rows']);
    end
    for r=1:3
        sl=subg*lengths(r,:);
        eval(['data=subj_-' gnames(r,:) '-']);
        data(:,2)=rand(sl,1);
        if sl<9
            dups=[data(:,1) rand(sl,1)]; %new set of random numbers
            dups=sortrows(dups,2);
            if sl>4
                data(L+1:9,:)=dups(sl-(9-sl)+1:sl,:);
            else %horrible
                data(:,2)=rand(sl,1);
                dups=data(:,1:4); %rand(sl,1)); %another new set of random numbers
                dups=sortrows(dups,2);
                data(9,:)=dups(4,:);
            end
        end
        data=sortrows(data,2);
        eval(['subj_-' gnames(r,:) '-data(:,1);']);
    end
    rowtrips=subj_free subj_dist subj_pace;
    for r=1:9
        for g=1:3
            leaveoutids(rowtrips(r,g),r)=1;
        end
    end
end

clabels=' hyp2';'hyp1_1';'hyp1_2';'hyp1_3';' hyp1';
for csl=1
    r02pr04Fadj=[r02pr04F leaveoutids];
    if csl==2
        %Groups 1 and 2
        findthhis=find(r02pr04Fadj(:,1)==20);
        r02pr04Fadj(findthhis,:)=[];%
        findthhis=find(r02pr04Fadj(:,1)==22);
        r02pr04Fadj(findthhis,:)=[];%
        findthhis=find(r02pr04Fadj(:,1)==25);
        r02pr04Fadj(findthhis,:)=[];%
        femalesnum=[11;12;16;18;19;15;17];
    elseif csl==3
        %Groups 2 and 4
        findthhis=find(r02pr04Fadj(:,1)==11);
    end
    trainandtestset=r02pr04Fadj;
    clear trainandtestclass;
    if csl==2
        for t=1:length(trainandtestset(:,1)); if trainandtestset(t,4)==1;trainandtestclass(t,1)=1;else;error('error in me');
        s(t,1)=1;elseif trainandtestset(t,4)==2;trainandtestclass(t,1)=-1;else;error('error in me');
        tup_d=1;for i=1:trainandtestset(t,:);tup_d=tup_d+1;end
        elseif csl==3
            for t=1:length(trainandtestset(:,1));trainandtestclass(t,1)=trainandtestset(t,4);%
        end
        trainandtestset=r02pr04Fadj;
        clear trainandtestclass;
        if csl==2
            for t=1:length(trainandtestset(:,1));if trainandtestset(t,4)==1 | trainandtestset(t,4)==2;
            trainandtestclass(t,1)=1;else;trainandtestset(t,4)=4;trainandtestclass(t,1)=2;
            end
            error('error in setup of matrices in trainvmresults');
        end
        trainandtestsetdata=[trainandtestset(:,1:15) trainandtestset(:,17:31) trainandtestse
        t(:,33:35) trainandtestset(:,45:47)];
        trainandtestset=trainandtestset(:,9) trainandtestclass trainandtestsetdata trainandtest
        set(:,57:65); leaveoutrows are at end
        eval([clabels(c,:)' _all'-'trainandtest']);
        for i=1:9
            trainandtestsubset=[trainandtest(:,1:28) trainandtest(:,28+4)];
            trainandtestsubset=sortrows(trainandtestsubset,29);
            findteststart=find(trainandtestsubset(:,29)==1);
            training=trainandtestsubset(1:findteststart(1)-1,:);
            testing =trainandtestsubset(findteststart(1):length(trainandtestsubset(:,1)),:);
            eval([clabels(c,:)' _C_10' num2str(i) '_traintesting']);
            eval([clabels(c,:)' _C_10' num2str(i) '_testtesting']);
            eval([clabels(c,:)' _C_10' num2str(i) '_[trainning;testing]');
        end
    end
end
save c:\sjm\Research\Matlab\classy\hypCmats hyp* leaveoutids

```

REFERENCES

- [1] Lemmon D; Shiang TY; Hashmi A; Ulbrecht JS; Cavanagh PR. The effect of insoles in therapeutic footwear -- a finite element approach. *J. Biomech*, 1997 Jun; 30(6): 615-20.
- [2] Morris ME; Huxham F; McGinley J; Dodd K; Iansek R. The biomechanics and motor control of gait in Parkinson disease. *Clin Biomech*, 2001 Jul; 16(6): 459-70.
- [3] Kerozek TW; LaMott EE; Dancisak MJ. Reliability of an in-shoe pressure measurement system during treadmill walking. *Foot Ankle Int*, 1996 Apr; 17(4): 204-9.
- [4] Zhu HS; Maalej N; Webster JG; Tompkins WJ; Bach-Y-Rita P; Wertsch JJ. An umbilical data-acquisition system for measuring pressures between the foot and shoe. *IEEE Trans Biomed Eng*, 1990 Sep; 37(9): 908-11.
- [5] Zhu HS; Wertsch JJ; Harris GF; Loftsgaarden JD; Price MB. Foot pressure distribution during walking and shuffling. *Arch Phys Med Rehabil*, 1991 May; 72(6): 390-7.
- [6] Zhu HS; Wertsch JJ; Harris GF; Alba HM; Price MB. Sensate and insensate in-shoe plantar pressures. *Arch Phys Med Rehabil*, 1993 Dec; 74(12): 1362-8.
- [7] Hausdorff JM; Ladin Z; Wei JY. Footswitch system for measurement of the temporal parameters of gait. *J Biomech*, 1995 Mar; 28(3): 347-51.
- [8] Hausdorff JM; Zemany L; Peng C.-K.; Goldberger AL. Maturation of gait dynamics: stride-to-stride variability and its temporal organization in children. *J Appl Physiol*, 1999 Mar; 86(3): 1040-7.
- [9] Hausdorff JM; Rios DA; Edelberg HK. Gait variability and fall risk in community-living older adults: a 1-year prospective study. *Arch Phys Med Rehabil*, 2001 Aug; 82(8): 1050-6.
- [10] Hausdorff, JM; Purdon PL; Peng C.-K; Ladin Z; Wei J.-Y.; Goldberger AL. Fractal dynamics of human gait: stability of long-range correlations in stride interval fluctuations. *J. Appl. Physiol*, 80: 1448–1457, 1996.
- [11] Morley RE Jr; Richter EJ; Klaesner JW, Maluf KS; Mueller MJ. In-shoe multisensory data acquisition system. *IEEE Trans Biomed Eng*, 2001 Jul; 48(7): 815-20.
- [12] Maluf KS; Morley RE Jr; Richter EJ; Klaesner JW; Mueller MJ. Monitoring in-shoe plantar pressures, temperature, and humidity: reliability and validity of measures from a portable device. *Arch Phys Med Rehabil*, 2001 Aug; 82(8): 1119-27.
- [13] Pappas IP; Popovic MR; Keller T; Dietz; Morari M. A reliable gait phase detection system. *IEEE Trans Neural Syst Rehabil Eng*, 2001 Jun; 9(2): 113-25.

- [14] Pappas IP; Keller T; Mangold S. A Reliable, Gyroscope based Gait Phase Detection Sensor Embedded in a Shoe Insole. Presented at the 2002 IEEE International Conference on Sensors, Orlando, FL.
- [15] Cutlip RG; Mancinelli C; Huber F; DiPasquale J. Evaluation of an instrumented walkway for measurement of the kinematic parameters of gait. *Gait Posture*, 2000 Oct; 12(2): 134-8.
- [16] Giacomozzi C; Macellari V. Piezo-dynamometric platform for a more complete analysis of foot-to-floor interaction. *IEEE Trans Rehabil Eng*, 1997 Dec; 5(4): 322-30.
- [17] See <http://www.cse.ohio-state.edu/~jwdavis/publications.html>
- [18] Yam CY, Nixon MS, Carter JN. Automated person recognition by walking and running via model-based approaches. *Pattern recognition*, 2004; 37(Part 5): 1057-1072
- [19] MIT AI Human ID Results [online]. Available at http://www.ai.mit.edu/people/llee/HID/mitai_data_avg_spec.htm [accessed 30 April 2004].
- [20] Johnson AY, Sun J, Bobick AF. Predicting large population data cumulative match characteristic performance from small population data. In the 4th International Conference on Audio- and Video-Based Biometric Person Authentication, Guildford, UK; June 9-11, 2003.
- [21] Walk the Walk: Gait Recognition Technology Could Identify Humans at a Distance; Georgia Tech Research News [online]. Available at <http://gtresearchnews.gattech.edu/newsrelease/GAIT.htm> [accessed 30 April 2004].
- [22] Kidd CD, Orr R, Abowd GD, Atkeson CG, Essa IA, MacIntyre B, Mynatt E, Starner TE, Newstetter W. The Aware Home: A living laboratory for ubiquitous computing research. In Proceedings of CoBuild '99: Second International Conference on Cooperative Buildings: 191-198.
- [23] Suutala J, Pirttikangas S, Riekki J, Röning J. Reject-optional LVQ-based two-level classifier to improve reliability in footstep identification.
- [24] The Miburi performance system, Yamaha Corporation, 1996 [originally available online, at "<http://www.yamaha.co.jp/news/96041001.html>"].
- [25] See <http://www.probalance.com>
- [26] Acceleron Technologies. The Leader in Smart Technologies for Health, Fitness and Beyond: Technology FAQ's [online]. Available at <http://www.xlrn.com/faq.html> [Accessed 22 February 2002].
- [27] See <http://www.traxtar.com>
- [28] See <http://www.fitsense.com>
- [29] See <http://www.vectrasense.com/>
- [30] Marriott M. The Bionic Running Shoe. *The New York Times*, May 6, 2004.

- [31] See <http://www.adidas.com/>
- [32] F-Scan System Features, Tekscan, Inc., 2004 [online]. Available from http://www.tekscan.com/medical/specs_fscan1.html [accessed 4 April 2004].
- [33] See <http://www.clevemed.com>.
- [34] Choi I, Ricci C. Foot-mounted gesture detection and its application in virtual environments. 1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation, Oct. 1997; 5(12-15):4248-53
- [35] <http://www.portablegaitlab.com/>
- [36] Acceleron Technologies, LLC. System and Method for Measuring Movement of Objects. United States Patent #5,899,963. 4 May 1999.
- [37] Acceleron Technologies, LLC. System and Method for Measuring Movement of Objects. United States Patent #6,122,960. 26 Sept 2000.
- [38] Paradiso JA; Hsiao K; Benbasat AY; Teegarden Z. Design and implementation of expressive footware. IBM systems journal, 2000; 39 (3): 511-519.
- [39] Krebs DE; Edelstein JE; Fishman S. Reliability of observational kinematic gait analysis. Phys Ther, 1985 Jul; 65(7): 1027-33.
- [40] Eastlack ME; Arvidson J; Snyder-Mackler L; Danoff JV; McGarvey CL. Interrater reliability of videotaped observational gait-analysis assessments. Phys Ther, 1991 Jun; 71(6): 465-72.
- [41] McGinley JL; Goldie PA; Greenwood KM; Olney SJ. Accuracy and reliability of observational gait analysis data: judgments of push-off in gait after stroke. Phys Ther, 2003 Feb; 83(2): 146-60.
- [42] Whittle MW. Clinical gait analysis: A review. Hum Mov Sci, 1996; 15 (3): 369-388.
- [43] Sutherland DH. The evolution of clinical gait analysis part 1:kinesiological EMG. Gait Posture, 2001 Jul; 14(1): 61-70.
- [44] Diss CE. The reliability of kinetic and kinematic variables used to analyse normal running gait. Gait Posture, 2001 Oct; 14(2): 98-103.
- [45] Fontaine D, David D, Caritu Y. Sourceless Human Body Motion Capture. Available online at <http://www.grenoble-soc.com/proceedings03/Pdf/30-fontaine.pdf> [accessed 9 May 2004].
- [46] Antonnsson EK. A three-dimensional kinematic acquisition and intersegmental dynamic analysis system for human motion. Ph.D. Thesis, MIT, 1982.
- [47] Riley PO; Mann RW; Hodge WA. Modelling of the biomechanics of posture and balance. J. Biomech, 1990; 23(5): 503-6.
- [48] Riley PO; Schenkman ML; Mann RW; Hodge WA. Mechanics of a constrained chair-rise. J Biomech, 1991; 24(1): 77-85.

- [49] Vicon Camera Technical Sheet, Vicon Motion Systems, 2003 [online]. Available at http://www.vicon.com/main/support/downloads/tech%20sheets/VMS027S_cameras_hi-res.pdf [accessed 27 October 2003].
- [50] Comparison Matrix: Eagle Digital, Hawk Digital, and Falcon Analog Systems, Motion Analysis Corporation, 2003 [online]. Available at <http://www.motionanalysis.com/pdf/systemcomp.pdf> [accessed 26 October 2003].
- [51] FASTRAK data sheet, Polhemus, 2001 [online]. Available at <http://www.polhemus.com/FastTrak/fastrak.pdf> [accessed 3 November 2003].
- [52] Raab FH, Blood EB, Steiner TO, Jones RJ. Magnetic position and orientation tracking system. *IEEE Transactions on Aerospace and Electronic Systems*, September 1979; AES15(5): 709-718.
- [53] Flock of Birds data sheet, Ascencion Technology Corp., 2000 [online]. Available at <http://www.ovation-tech.com/products/flockofbirds.pdf> [accessed 3 November 2003].
- [54] Hausdorff JM; Ladin Z; Wei JY. Gait variability and basal ganglia disorders: stride-to-stride variations of gait cycle timing in Parkinson's disease and Huntington's disease. *Mov Disord*, 1998 May; 13(3): 428-3.
- [55] Thaut MH; Kenyon GP; Schauer ML; McIntosh GC. The connection between rhythmicity and brain function. *IEEE Eng Med Biol Mag*, 1999 Mar-Apr; 18(2): 101-8.
- [56] Thaut MH; McIntosh GC; Rice RR, Miller RA, Rathbun J, Brault JM. Rhythmic auditory stimulation in gait training for Parkinson's disease patients. *Movement Disorders*, 1996; 11(2): 193-200.
- [57] McIntosh GC; Rice RR; Hurst CP, Thaut MH. Long-term training effects of rhythmic auditory stimulation on gait in patients with Parkinson's disease. *Movement Disorders*, 1998; 13(suppl 2): 212.
- [58] Pacchetti C; Mancini F; Aglieri R; Fundaro C; Martignoni E; Nappi G. Active music therapy in Parkinson's disease: an integrative method for motor and emotional rehabilitation. *Psychosom Med*, 2000 May-Jun; 62(3): 386-93.
- [59] Hale LA; Eales CJ. Recovery of walking function in stroke patients after minimal rehabilitation. *Physiother Res Int*, 1998;3(3):194-205.
- [60] Bessou P; Dupui P; Montoya R; Pages B. Simultaneous recording of longitudinal displacements of both feet during human walking. *J Physiol*, 1988-89; 83(2): 102-10.
- [61] Ebersbach G; Sojer M; Müller J; Heijmenberg M; Poewe W. Sociocultural differences in gait. *Mov Disord*, 2000 Nov; 15(6): 1145-7.
- [62] S700 Joint Angle Shape Sensor data sheet, Measureand Inc., 2004 [online]. Available at <http://www.measurand.com/manuals/S700.pdf> [accessed 4 April 1004].
- [63] Baxter LK. Capacitive Sensors. IEEE Press, New Jersey, 1997.

- [64] As cited in Hausdorff JM; Zemany L; Peng C.-K.; Goldberger AL. Maturation of gait dynamics: stride-to-stride variability and its temporal organization in children. *J Appl Physiol*, 1999 Mar; 86(3): 1040-7.
- [65] Benbasat AY, Morris SJ, Paradiso JA. A wireless modular sensor architecture and its application in on-shoe gait analysis. Proc. of the IEEE International Conference on Sensors, Oct. 21-24 2003; p 1086-1091.
- [66] Benbasat AY. Stack: A Modular Platform for High Density Wireless Sensing [online]. Available from <http://www.media.mit.edu/resenv/Stack/> [accessed 22 February 2002].
- [67] Winter DA, Sidwall HG, Hobson DA. Measurement and reduction of noise in kinematics of locomotion. *J Biomech*, 1974; 7:157-159.
- [68] Winter DA. Camera speeds for normal and pathological gait analyses. *Med Biol Eng Comput*, 1982; 20:408-412.
- [69] Winters DA. Biomechanics and motor control of human movement. John Wiley & Sons, 1990, 2nd ed.
- [70] Weinberg, Harvey. MEMS (Micro Electro-Mechanical Systems) Technology. [online]. Available at <http://www.sensorland.com/HowPage023.html> [accessed 12 December 2003].
- [71] ADXL202E Data Sheet, Rev A, Analog Devices, Inc., 2000 [online]. Available at http://www.analog.com/UploadedFiles/Data_Sheets/70885338ADXL202_10_b.pdf [accessed 29 May 2002].
- [72] Yazdi N, Ayazi F, Najafi K. Micromachined inertial sensors (invited paper). *Proceedings of the IEEE*, 1998; 86(8):1640-1659.
- [73] Murata ENC-03J Specifications [online]. Available from <http://search.murata.co.jp/Ceramy/owa/CATALOG.showcatalog?sHinmTmp=ENC-03J&sLang=2&sNhm=ENC-03J&sHnTyp=OLD> [accessed 28 October 2002].
- [74] Krakauer D. A unique angular-rate-sensing gyro [online]. *Sensors Magazine*, Dec 16, 2003. Available at <http://www.sensorsmag.com/articles/0903/53/main.shtml> [accessed 12 December 2003].
- [75] ADXRS150 Data Sheet, Rev A, Analog Devices, Inc., 2003 [online]. Available at http://www.analog.com/UploadedFiles/Data_Sheets/778386516ADXRS150_B.pdf [accessed 15 April 2003].
- [76] Fraden J. *Handbook of modern sensors: physics, designs, and applications*. Springer Verlag, 1997, 2nd ed.
- [77] FSR integration guide and evaluation parts catalog, with suggested electrical interfaces, Interlink Electronics [originally available online, from <http://www.interlinkelec.com>].
- [78] Images Co. FLX-01 Specifications [online]. Available from <http://www.imagesco.com/catalog/flex/FlexSensors.html> [accessed 22 February 2004].

- [79] American Piezo Ceramics. Piezoelectric Ceramics: Principles and Applications. APC International, Ltd. [year to be added]
- [80] Piezo Film Sensors Technical Manual, Measurement Specialties [online]. Available at http://www.msiusa.com/piezo_download_listing.htm [accessed 8 April 2004].
- [81] Ohr, S. Non-contact sensor discerns passenger size for airbags. EE Times, February 6, 2003 [online]. Available at <http://www.eetuk.com/tech/news/OEG20030206S0035> [accessed 4 April 2004].
- [82] MC3394 data sheet, Rev 6.0, Motorola, Inc., Feb 2003 [online]. Available at http://e-www.motorola.com/files/analog/doc/data_sheet/MC33794.pdf [accessed 17 December 2003].
- [83] Burke SE, Paradiso JA. High-Resolution Piezopolymer Acoustic Bearing Estimator. Presented at the Second Technical Conference on Telecommunications R&D in Massachusetts, University of Massachusetts, Lowell MA, March 12, 1996.
- [84] Coon A. Capabilities and Applications of SAW Coupled-Resonator Filters (AN 23), RF Monolithics, Inc., [online]. Available at <http://www.rfm.com/corp/appdata/An23.pdf> [accessed 30 April 2004].
- [85] DR3000-1 Data Sheet, RF Monolithics, Inc., [online]. Available at <http://www.rfm.com/products/data/dr3000-1.pdf> [accessed 24 July 2002].
- [86] Titterton D, Weston J. Strapdown Inertial Navigation Technology. IEEE, 1997.
- [87] Brown RG, Hwang PYC. Introduction to Random Signals and Applied Kalman Filtering with Matlab Exercises and Solutions. Wiley Text Books, 1996, 3rd ed.
- [88] Allen BD, Bishop G, Welch G. Tracking: Beyond 15 Minutes of Thought. 2001 SIGGRAPH Course Notes, Course 11.
- [89] Kuipers JB. Quaternion and rotation sequences: a primer with applications to orbits, aerospace, and virtual reality. Princeton University Press, 1999.
- [90] Hogan N, Krebs D, et al. MIT 2.75s Course Outline; June 19-23, 1995.
- [91] Lawrence A. Modern inertial technology: navigation, guidance, and control. Springer-Verlag, 1998.
- [92] Insole pressure data during gait, collected using the Tekscan F-Scan; received via e-mail communication on 19 Apr 2004 with Mike Harty and Thomas Papakostas at Tekscan, Inc.
- [93] Breiman L, Friedman J, Olshen R, Stone C. Classification and Regression Trees. Pacific Grove: Wadsworth, 1984.
- [94] Steinberg D, Colla P. CART – Classification and Regression Trees. San Diego, CA: Salford Systems, 1997.
- [95] Bayes, Rev. T., "An Essay Toward Solving a Problem in the Doctrine of Chances", Philos. Trans. R. Soc. London, 1763, 53:370-418.

- [96] Duda RO, Hart PE, Stork DG. *Pattern Classification*. Wiley-Interscience, 2000, 2nd ed.
- [97] Witten IH, Frank E. *Data Mining: Practical Machine Learning Tools and Techniques with Java IMplementations*, Chapter 8: WEKA Machine Learning algorithms in Java. Morgan Kaufmann Publishers, 2000.
- [98] Boser B, Guyon IM, and Vapnik VN. A Training Algorithm for Optimal Margin Classifiers, Computational Learing Theory 1992.
- [99] Support Vector Machines: <http://www.support-vector.net/>
- [100] See <http://www.isis.ecs.soton.ac.uk/resources/svminfo/>
- [101] Gunn SR. *Support Vector Machines for Classification and Regression*. Technical Report, Image Speech and Intelligent Systems Research Group, University of Southampton, 1998.
- [102] Cristianini N, Shawe-Taylor J. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.
- [103] Reed RD, Marks RJ. *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*. MIT Press, 1999.
- [104] Neural Network Toolbox User's Guide. The MathWorks, Inc. [online]. Available at http://www.mathworks.com/access/helpdesk/help/pdf_doc/nnet/nnet.pdf [accessed 25 March 2004].
- [105] Perry J. *Gait Analysis: normal and pathological function*. SLACK Inc, 1992, 1st ed.
- [106] Theodoridis S, Koutroumbas C. *Pattern Recognition*. Academic Press, 2003, 2nd ed.
- [107] Paradiso JA, Morris SJ, Benbasat AY, Asmussen E. Interactive Therapy with Instrumented Footwear. Proceedings of CHI 2004, April 24-29, 2004, Vienna, Austria.
- [108] Wigram T, Nygaard Pedersen I, Ole Bonde L. *A Comprehensive Guide to Music Therapy: Theory, Clinical Practice, Research and Training*. Jessica Kingsley Publishers, London, England, 2003.
- [109] Rosenboom D (ed.). *Biofeedback and the Arts: Results of Early Experiments*. A.R.C. Publications, Vancouver, 1976.
- [110] Gromala D, et al. *Meditation Chamber*. SIGGRAPH 2001 Conf. Abstracts and Applications, ACM Press, 2001; p. 127.
- [111] Ellis P. The music of sound: a new approach for children with severe and profound and multiple learning difficulties. *British Journal of Music Education*, 1997; 14(2): 173–186.
- [112] Gerasimov V. *Swings That Think* [online]. Available from <http://vadim.www.media.mit.edu/stt/bat.html> [accessed 4 April 2004].
- [113] Asmussen E. *An Auditory Biofeedback Program for Gait Analysis*. Responsive Environments Group Technical Report, MIT Media Lab, August 2003.

- [114] See <http://www.cycling74.com/products/maxmsp.html>
- [115] See <http://www.keyspan.com/products/usb/usa19hs/>
- [116] See <http://www.midiman.com/>
- [117] See <http://www.emu.com/>
- [118] See <http://www.pdf.org/>
- [119] See <http://www.parkinson.org/>
- [120] See http://www.msiusa.com/piezo/ultrasonic_transducers.htm
- [121] See <http://www.gibsontech.net/vma40a5r.html>