Pressure sensor system for investigating stresses at the stump/socket interface.

Roberto Moratore Jr
S2257017

Project supervisors:
Corry van der Sluis
Raoul Bongers

Thesis coordinator:
Bart G.J. Verkerke

August, 2014

Table of Contents

Abstract 3

Background 4

Introduction 5

Sensor Design 6
- **Analysis Phase** 6
  - Problem definition 6
  - Design assignment 6
    - Restrictions 6
  - Requirements and Wishes 7
    - Function 7
    - Dimension 7
    - Cost 7
    - Safety 7
  - Function analysis 8
- **Synthesis** 9
    - Brainstorm 9
    - Final Concept 12
    - Prototype 14
    - Prototype Testing 15

Experiment 16
- **Methodology** 16
  - Participants 16
  - Sensor Placement 17
  - Procedure 18
- **Analysis** 20
- **Results** 20
- **Experiment Discussion** 24

Conclusion 25

Recommendations 25

References 26

Appendix 27
- **Appendix 1 – Data Collection** 27
- **Appendix 2 – Data Analysis** 32
    - Repetition Correlation 32
    - Data Resampling 33
    - Peak finding algorithm 35

Abstract

Upper-limb deficient persons (UDL) often report musculoskeletal and stump pain after prolonged usage of prosthetic devices. One possible cause is the stress between socket and stump. This stress has never been investigated in literature. In this project, a system of pressure sensors was developed using Force Sensing Resistors in order to measure this stress during activities of daily living (ADL). This project is being done in partnership with an existing project by the Department of Rehabilitation Medicine at the UMCG, whereupon the effects of flexible wrists on socket pressure are being investigated. Due to this partnership, the study design from their project has been emulated here, and pressure differences between flexible and static wrists were compared.

Eight participants were measured performing six different ADLs. Participants were measured while wearing the Otto Bock® Flex-Wrist (OB) (-40, -20, 0, 20, 40 flexion/extension) and Motion Control's® Multi-flex wrist (MC) (-30, 0, 30, free flexion/extension with free radial and ulnar deviation) with corresponding hands. These wrists allow users to select different degrees of flexion or extension. The measurements were done in a static condition, where the wrists were set in the middle setting, as well as a flexible condition where the participants were allowed to choose the position of the wrist freely.

The design was proved to be successful and pressure was measured with signinficant repetability. Results showed no difference in the pressure between flexible and static wrists. However MC had higher mean and maximum pressures than OB in some participants, but no apparent reason for this was observed. It can be concluded that flexible wrists offer no advantages over static wrists when it comes to the pressure in the socket, but further research to confirm this conclusion is needed.

Background

Musculoskeletal and stump pain complaints are often reported by upper limb prosthetic users (Ephraim et al., 2005). These complaints have been related to the necessity of users to perform compensatory movements (Ostlie et al., 2011) (Metzger et al., 2012). The inevitability for compensatory movements arises when the degrees of freedom (DOF) of the prosthesis in general and the prosthetic wrist in particular are limited. Traditionally, upper limb myoelectric prostheses have only one DOF (rotation), therefore proximally located joints (e.g. elbow, shoulder, and trunk) ultimately control hand positioning (Bertels et al., 2009) (Carey, et al., 2008). With this in mind, more and more current prosthetic hands come equipped with wrists that not only rotate, but also can flex or extend and in some cases allow for radial or ulnar deviation. Henceforth wrists with only one DOF (rotation) shall be referred to as "static wrists" and wrists with higher DOF as "flexible wrists."

Limited evidence exists in literature in support of efficacy of flexible wrists diminishing compensatory movements (Bertels et al., 2009), as well as increasing functionality (Zinck et al., 2012). These studies, however, either had a much too small sample size or did not measure enough outcomes. Furthermore, user satisfaction of flexible wrists as compared to static wrists has never been evaluated. In light of this, Marieke Deijs, under the supervision of Corry van der Sluis and Raoul Bongers at the Department of Rehabilitation Medicine at the University Medical Center Groningen (UMCG), is performing a study to determine the efficacy of flexible wrists compared to static ones across many parameters. One such parameter is whether the pressure inside the socket will change when performing certain tasks between the flexible and static wrists. This project will concentrate on the methodology for measuring such pressure, as well as analyzing the measurement results.

Introduction

This project is a collaboration between Biomedical Engineering Department and the Department of Rehabilitation at the UMCG. It is a hybrid between a design assignment and a research project. The first part of this thesis describes in detail the process of designing and testing the system to measure the pressure inside the socket. The second part then deals with the study design for measuring the pressure, as well as data analysis comparing the pressure for the flexible and static wrists.

In the design section, a rational for all design choices is given by describing each step of the design process. First, the analysis phase is explained, where the general characteristics of the design are delineated, together with limitations, which guided the decisions. Second, the main choices for the design are explored with all benefits and disadvantages explained. Third, the final concept is detailed and the prototype is discussed, as well as the testing process used to optimize it.

The study concentrates on the efficacy of two flexible wrists, namely the Flex-Wrist by Otto Bock ® and the Multiflex wrist by Motion Control ®. These wrists prove ideal to evaluate pressure differences, as they are equipped with conventional prosthetic hands. Other flexible prosthesis (e.g. i-Limb Ultra) are equipped with more advanced hands with multiple grip patterns, and thus it is more difficult to correlate the effects to wrist differences.

Otto Bock's ® Flex-Wrist allows users to lock the wrist at 20° or 40° flexion, a neutral position at 0°, or 20° or 40° extension. Different position can be selected by pressing a button on the side of the wrist, which unlocks the hinge. Then the user can move the hand to the desired position and then release the button to once again lock the hinge. This wrist comes equipped with the Otto Bock ® Transcarpal Hand.

Motion Control's ® Multiflex wrist allows users to lock the wrist at 30° flexion, a neutral position at 0°, or 30° extension also by unlocking the hinge with a button on the side of the wrist. Users also have the option of leaving the wrist unlocked at 0°, where the wrist remain flexible to flexion or extension. In both locked and unlocked modes, the wrist flexible in the radial and ulnar directions. This wrist comes equipped with Motion Control's ® ProPlus Hand.

Participants were grouped into two groups according to stump length (long - >15cm, short <15cm) to isolate the effect of the wrists. Pressure is a function of surface area. As all participants performed the same task, similar loads were applied to their prosthesis. It stands to reason that participants with shorter limbs had a higher pressure on their limbs.

The thesis is concluded with a discussion about the design of the prototype to measure stress inside the socket of an arm prosthesis, and an exploration of the limitations of the design. Finally recommendations on how to improve the design in future studies will be made.

Sensor Design

Analysis Phase

In the analysis phase, the problem to be solved is defined, goals and restrictions are set, and the desired features for the solution are outlined. This will serve as a base to judge ideas that arise in the brainstorming session.

Problem definition

Upon literature review, no previous study was found where the pressure inside the socket for upper limb prosthesis was measured. Some literature (Convery & Buis, 1998) (Sanders & Daly, 1993) was found where the pressure in the socket of lower limb prosthesis was measure. The methods used however do not directly translate to upper limb prosthesis, as the load distribution and space inside the socket are completely different. In upper limb prosthesis the load is applied radially, where in lower limb the load is axial. Thus for the lower limb, it is enough to place sensors on the bottom of the sockets, where for upper limb sensors have to be placed all the way around. Furthermore, upper limb socket form a tight fit with the stump, and virtually no space is available to place the sensors. Therefore, the main problem to be addressed is how to overcome space limitations to measure pressure in the socket of upper limb prosthesis.

Design assignment

Before we can properly establish the design assignment, some very important restrictions must be made to guide our design.

Restrictions

There are two main features that guided the design process and must be carefully examined: the maximum thickness of the sensor and the pressure range the sensor must measure. As no literature is available where this information can be derived from, tests and assumptions must be made.

To determine the maximum thickness of the sensor, mock sensors of different thicknesses (0.5mm, 1mm, 2mm, 3mm, and 4mm) were made. Each sensor was then taped to the stump of volunteers one at a time. The volunteers were then asked whether they could feel the sensor, and if so, whether it was uncomfortable. It was also noted how easily the volunteers were able to put their prostheses on. This was done with four volunteers, and in all cases it was found that 3mm was the maximum thickness where volunteers were still able to put their prostheses on and it was not uncomfortable.

To determine the pressure range the sensor must work in, the pain pressure threshold (PPT) was used. The PPT is the amount of pressure needed to cause pain. Literature was found that showed an average of 230 kPa PPT in the forearm (Jones et al., 2007). As prosthesis users are not in constant pain due to the pressure of the socket, an upper range of 200 kPa was decided upon.

Requirements and Wishes

The requirements and wishes are based on the previously defined problem as well as in the define restrictions. Requirements are features the final device must have, where wishes are features that are desirable but not completely necessary.

Function

The final device should be easy to implement and should be able to record forces without any interference with the prosthesis.

Requirements

1. The device must measure and record forces over time.
2. The device must measure pressure.
3. The device must be able to measure 0-200 kPa.
4. The device must connect to a computer, where the data can be displayed both visually and numerically.

Wishes

1. The device should be wireless.
2. The device should measure shear force.

Dimension

The dimension requirements are based on the data obtained from the measurements done with participants of the study.

Requirements

1. The device must be flexible enough to fit any stumps.
2. The device should not be thicker than 3 mm

Wishes

1. Optimally the device should be almost imperceptible to the user.

Cost

These costs are based on the limited funding from the partnering research group as well as other external resources.

Requirements

1. The device should not exceed €250.00

Safety

Considering the non-invasive nature of the device, the safety requirements are minimal.

Requirements

1. The device must not cause any additional pain or discomfort.

Function analysis

In order to measure the pressure in the socket, sensors must be used to convert mechanical energy (pressure) into electrical energy (voltage). The voltage is an analog signal, and must be converted into an electrical signal in order for it to be received, stored, and subsequently analyzed by the computer. Both conversions (pressure to analog signal and analog to digital) require energy. A function diagram can be seen in Figure 1.

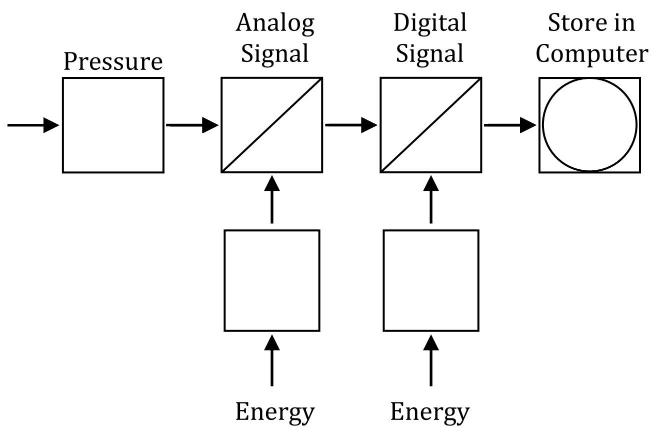

Figure 1 - Function Analysis.

Synthesis

In the synthesis phase, all the definitions placed in the analysis phase are turned into real solutions, which eventually lead to a final design. First, a broad range of ideas is created by brainstorming. It is important to note that the feasibility of the idea is not important in the beginning of the process, as even unfeasible ideas might lead to good solutions. By the end of the process however, the most feasible ideas are selected and compared against the requirements set in the previous section, and a final concept is chosen.

Brainstorm

When a broad problem is posed, a brainstorming session is ideal to lay down a base of ideas to build up from. However as this project has a very specific goal, with very well defined subfunctions, a morphological map is used to create pre-concepts.

Morphological Map

As seen from the function analysis, there are five subfunctions of the design that need to be solved. First, an appropriate sensor needs to be chosen to convert the pressure into an analog signal. Second the analog signal needs to be converted into a digital signal to be interpreted by the computer. Third, the digital signal needs to be transmitted to the computer. Fourth, an energy storage solution needs to be found to power the system. Finally, a computer interface must be created to store and display the acquired data. Considering the scope of this project, as well as time limitations, it is unpractical to develop everything from scratch; therefore existing solutions for each subfunction were found and combined to form the final design. Table 1 details the morphological map.

Table 1 - Morphological map.

Function	Idea 1	Idea 2	Idea 3	Idea 4
Sensor	Custom Woven Sensor Matrix	SPI Tactilus Stretch Sensor	University of Twente custom sensor	Force Sensing Resistor
Digital Converter	Custom IC	Proprietary SPI Package	Arduino	
Transmission	Wi-Fi	Bluetooth	USB	Serial Port
Energy Storage	Battery	USB		
Interface	MATLAB® R2012B	Proprietary SPI Package	SensoDuino	

Sensors

Custom Woven Sensor Matrix

These sensors are created by weaving layers of conductive fabric, non-conductive fabric, and a pressure-sensitive conductive sheet (i.e. Velostat). Each intersection of the weave creates one sensor; therefore, as the size of the weave diminishes, the resolution increases. Figure 2 shows a sample weave.

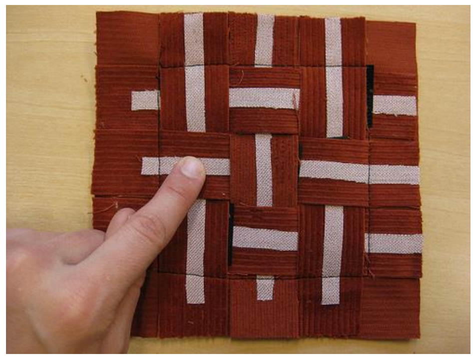

Figure 2 - Sample weave (Satomi & Wilson, 2013)

This sensor has the advantage of being highly customizable, and being adaptable enough to fit any stump size. It however, is not very well documented, and extensive tests would need to be conducted before the properties of this sensor could be determined.

SPI Tacitus Stretch Sensor

This is similar to the woven sensor matrix, however individual fibers are used for the weave to create the sensors. This creates a much thinner and higher resolution system. Furthermore, this sensor is extremely stretchable and can easily fit any stump. It can measure from 0 to 200 kPa, thus being exactly in the required range. Unfortunately, the sensor prices start at around €9000, going extremely over budget.

University of Twente custom Sensor

The University of Twente is currently developing a new sensor, which can measure both pressure and shear force (the only one from the current selection). It can measure from 0 to about 500 kPa, encompassing the necessary range. Furthermore, it is only 1 mm thick being perfect for this application. Finally this sensor would be free as it would be a collaboration between universities. Unfortunately, the sensors were not ready in time to be used by this project.

Force Sensing Resistor

Force Sensing Resistors (FSR) are resistors, which are susceptive to an applied load. When unloaded, they have a resistance of 10MΩ, which goes down to about 100Ω when loaded. They can measure a pressure range between 0 to around 700 kPa. Furthermore they are only 0.46 mm thick and around €4 per sensor, being perfect for this application. Unfortunately they have about a 15% error margin when measuring, and cannot measure shear forces.

Digital Converters

Custom Integrated Controller

This is a very broad category. There are many integrated controllers on the market that can convert analog to digital signals. These are relatively cheap. The drawback with these is that the supporting circuitry needs to be design and built and the programing varies depending on the vendor.

Proprietary SPI Package

This is a proprietary package developed by SPI to be specifically used with their sensors.

Arduino

Arduino is a single board microcontroller, which is excellent for prototyping. The programming language it uses is similar to C and documentation is readily available. This is the simplest option to use, as minimal customization is needed. Furthermore, it costs only about €20.

Interface

Here the programs used to record and display sensor data are briefly described.

MATLAB

MATLAB® R2012B is a numerical computing environment, which can interface with other software and different hardware. A highly customizable interface can be created for data capture and display. It also provides robust resources for data manipulation. Finally, extensive documentation is readily available.

Custom SPI Package

This is a proprietary package developed by SPI to be specifically used with their sensors.

SensoDuino

SensoDuino is a package for the Android platform, which interfaces with Arduino. It allows the user to send commands to the Arduino, as well as capture sensor data to a comma separated value file. Although it allows the use of a small mobile platform for data capture, it is not customizable and has a poor interface.

Final Concept

The final concept consists of the Force Sensing Resistors connected to an Arduino Micro. This is both powered and connected to the computer over USB. The software of choice was MATLAB® R2012B. This proved to be the cheapest most customizable option available given the time and budget restrictions.

Twelve Interlink Electronics Model 402 force sensing resistors (FSRs) were connected to an Arduino Micro ®. The FSRs were built from a printed semi-conductor layer separated by a spacer from a printed electrode layer (Figure 3). As pressure is applied the top and bottom layer come in contact and the resistance diminishes. Unloaded, the FSRs have a resistance greater than 10MΩ. When a minimum of 0.2N is applied to the sensor, resistance decreases almost linearly as given by Figure 4.

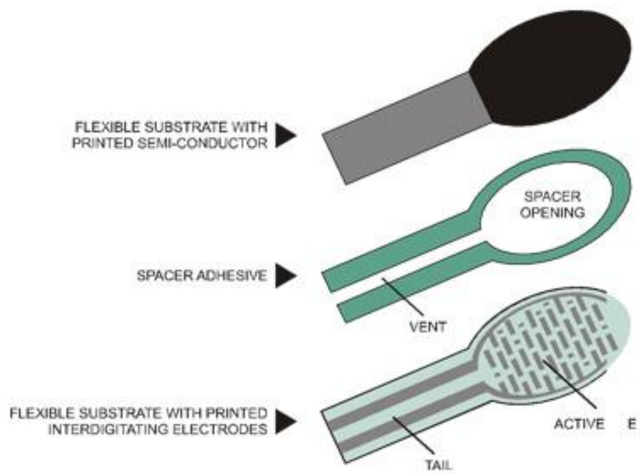

Figure 3 - FSR diagram (Interlink Electronincs)

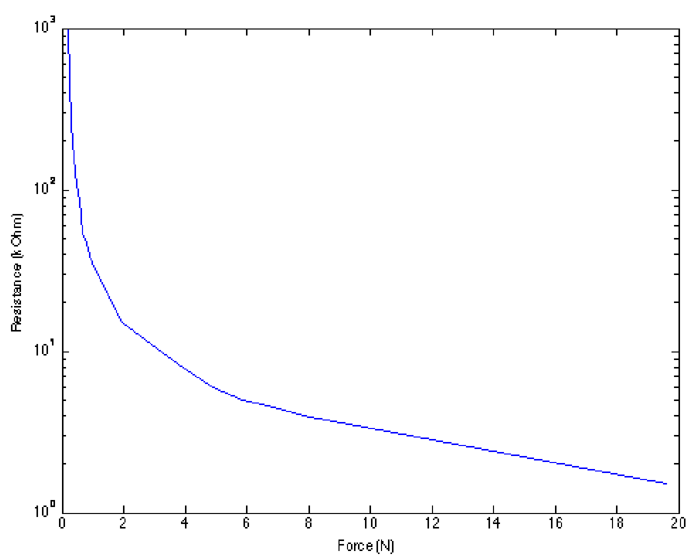

Figure 4 - Resistance/Force Curve

The FSRs are connected to an Arduino Micro through a simple voltage divider circuit (Figure 5). This circuit uses a second reference resistor (10kΩ) to change the resistance value of the FSRs into a fraction of the input voltage (5V), which can be read by the Arduino. 10kΩ reference resistors were chosen as they produce a relative linear voltage curve at different forces, while taking advantage of a large voltage range (~1-3.5V) Figure 6.

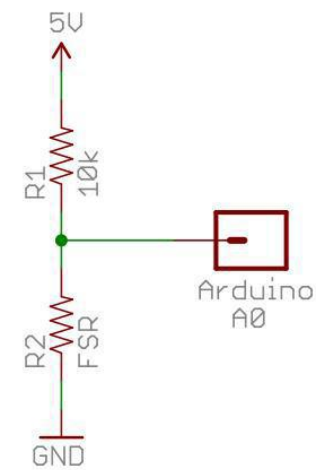

Figure 5 - Voltage divider diagram

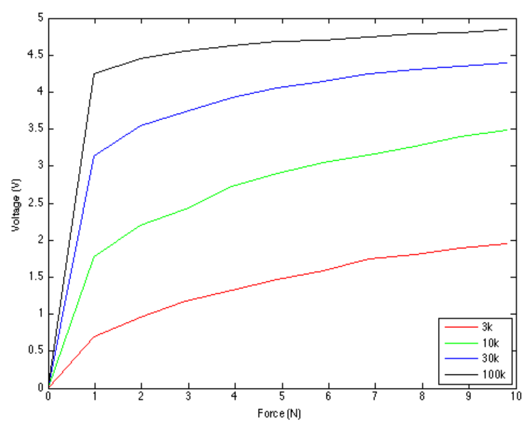

Figure 6 - Voltage over force curves for different reference resistor values

The sensors were connected in two arrays of four and two arrays of two. These arrays can be combined in any order, allowing for the use of two to twelve sensors in increments of two. The sensors are connected to the Arduino using S-Video connectors.

The Arduino is then serially connected over micro-USB to a computer running MATLAB® R2012B. A script (Appendix 1) was written to calibrate each FSR by recording the voltage output when 0 to 1000g were applied in 100g increments and a curve created from these values and saved. Another script reads the output voltages from the Arduino, and using the curve obtained from the calibration and the sensing area of the FSR calculates and stores the measured pressure in Pascals.

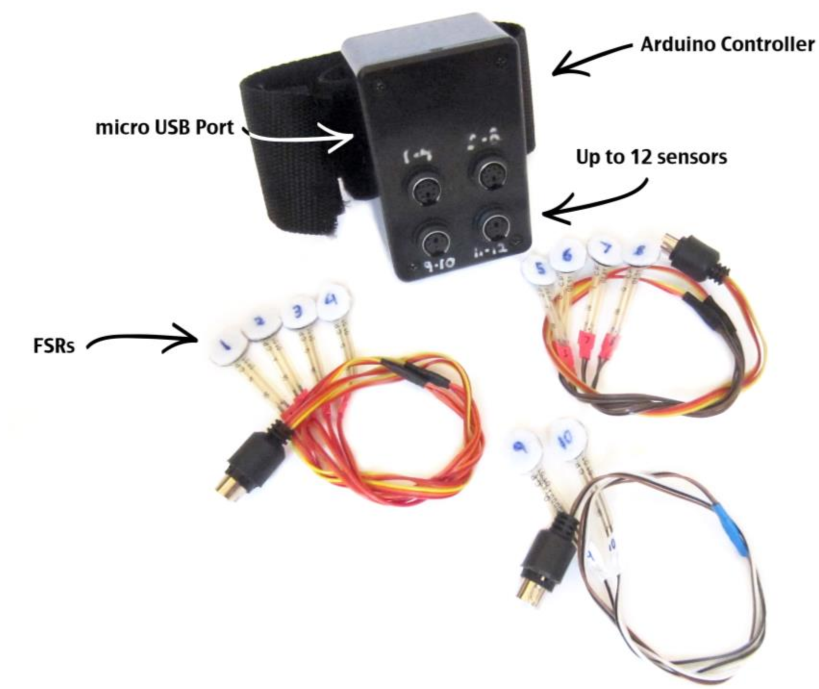

Figure 7 – Prototype (three out of four sensor arrays shown). The black box contains the Arduino controller that is connected to the computer with a micro USB port indicated in the figure. The sensor arrays (indicated as FSRs) are connected to the four connections on the front of the box. Up to four arrays (two four-sensor arrays, and two two-sensor arrays) can be connected to the box, giving a range of two to twelve sensors for measurement.

In the final prototype, all electronics were placed in an 85 x 55 x 40 mm box with a Velcro strap attached to the back. The front of the box had four ports where the sensor arrays are connected. The first two ports connect to the two four sensor arrays and the bottom two to the two sensor arrays. A micro USB port is located on the right side of the box.

When measuring, first the sensors are taped to the participants arm. The box is then attached to the participant's arm with the Velcro straps, and the sensors connected to the ports in front of the box. Finally, a 4.8 m cable is connected to the micro USB port on the box and a USB port in the computer. The MATLAB® R2012B script is started, and the sensors are individually tested by applying pressure on them one at a time. Once the sensors are confirmed to be working, the participant puts on the prosthesis, and the sensors are again checked to be working by applying a torque to the prosthesis. Once all sensors are confirmed to still be working, the measurement starts.

To test the prototype at every stage of construction, a dummy stump was built by making a mold of the inside of a prosthetic socket by using liquid latex. This mold was then filled with a gelatin based ballistic gel to simulate the consistency of human flesh. A wooden dowel was inserted in gel to simulate bones. The sensors were then taped on the mock stump, and the stump inserted into the socket. With this set up, the optimal cable layout (to minimize the number of cables, while still allowing for flexibility in sensor placement) was found.

To ensure reproducibility, a 1 kg weight was placed on each sensor twenty times, and the output voltage was recorded. In between each measurement, the sensors were mounted on the mock stump, place in the socket, and the stump moved in the socket to simulate activity. On average there was a 4.14% variance between the maximum and minimum values. Figure 8 shows the values of individual measurements per sensor. These measurements were used to check the error of the reported weight when compared to the placed weight. On average, 1.17 kg was reported, or 16.58% error.

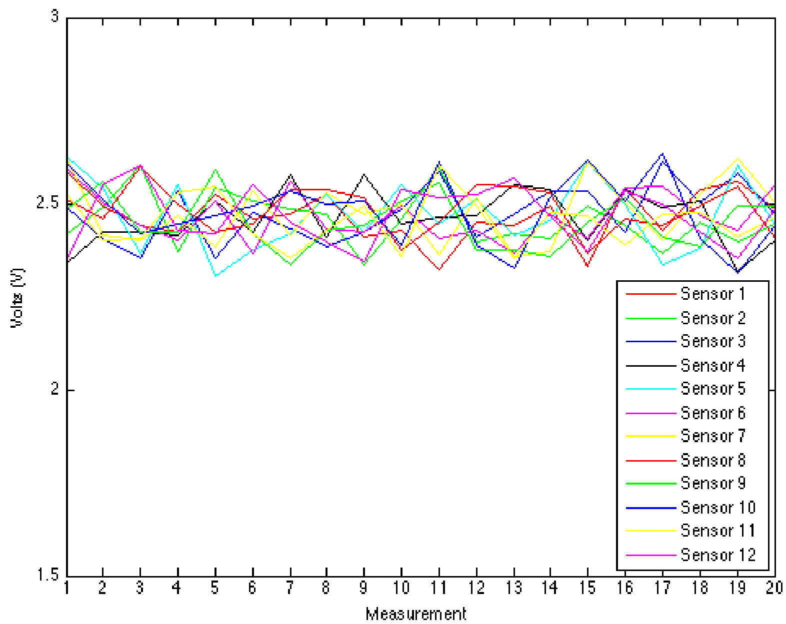

Figure 8 - Measurement reproducibility

When the prototype was completed, a pilot measurement was done with a participant to ensure the design would work with a real stump. Once the sensors were placed, and the participant put on his prosthesis, no pressure was registered. It was concluded that the sensors were either too thin, or due to the design of the sensor, all the load was applied on the edges of the sensor (where the spacer is located) thus not allowing the two parts of the FSR to come in contact. To remedy this, 1 mm felt disks were glued to the sensor. Once this was done, pressures were registered without a problem.

This pilot measurement was also used to optimize the MATLAB® R2012B script, which recorded, and displayed the sensor data. Originally the script recorded all four repetitions of each task together. However this was found to be cumbersome and difficult to deal with for data analysis. Each repetition was recorded individually instead.

Experiment

Methodology

The experiment design was carried over from the partner study.

Participants

All but two participants carried over from the partner study. Originally, participants were selected from the orthopedic workshop OIM Orthopedics at the UMCG and Revant Rehabilitation Center in Breda patient databases. Participants were considered eligible if they met the following five conditions; (1) Participants had a minimum of one year experience with myoelectric forearm prosthesis with wrists that can only rotate; (2) participants wore their prosthesis for a minimum of four hours a day; (3) participants had hands with only one DOF (opening and closing); (4) participants had no experience with either the Flex Wrist or Multiflex wrist; (5) participants were free of co-morbidities which could hinder the tests (e.g. neurological disorders or rheumatic diseases that diminished arm function). Eight participants (six men, two women; 54 ± 14 years) were included in the experiment. Two participants from the original experiment were unable to participate in this experiment, and were replaced by participants from the same databases. Participant Characteristics are shown in Table 2.

Table 2 – Participant Characteristics

	Gender	Age (years)	Cause of disability	Time after amputation (years)	Experience with prosthesis (years)	Stump length
P1	M	43	ULA	10	10	21 cm
P2	M	44	ULRD	N.A.	22	18 cm
P3	M	49	ULA	21	>7	13 cm
P4	M	54	ULA	12	10	27 cm
P5	M	55	ULA	22	22	9.5 cm
P6	M	71	ULA	50	30	23.5 cm
P7	F	62	ULRD	N.A.	5	12 cm
P8	F	65	ULRD	N.A.	55	11 cm

Abbreviations: M – male; F – female; ULRD – upper limb reduction deficiency; ULA – upper limb amputation; N.A. – Not applicable.

Participants received a €200 gift voucher in the original study, and travel compensation both for the original experiments and the current study. Participants provided written informed consent. The study was approved by the Medical Ethical Committee of the University Medical Center Groningen (NL44256.042.13).

Sensor Placement

A total of ten sensors were available for placement. For P1 to P4 all ten sensors were used. Unfortunately the measurements for P5 to P8 were performed in close succession and five sensors broke during the fitting of P5, therefore only five sensors were used during those measurements. All sensors were placed in similar areas for all participants (P5 to P8, corresponding to sensors 1 through 10 of P1 to P4). The sensors were simply taped to the participants arm, ensuring that none overlapped the location of the electrodes to ensure proper functioning of the prosthesis. Table 3 shows the locations the sensors were placed. The sensors closest to the elbow are labeled as 'Elbow', those in between the elbow and the wrist labeled 'Mid', and those closer to the wrist labeled 'Wrist.' Four sensors were place around the arm (anterior, ulnar, posterior and radial sides) at the 'Elbow' and 'Mid', and two sensors (anterior and posterior) at the 'Wrist.' Figure 9 shows a sample sensor placement.

Table 3 - Sensor placement

	Elbow	Mid	Wrist
Anterior	1	5	9
Ulnar	2	6	
Posterior	3	7	10
Radial	4	8	

Numbers indicate sensor id, and not quantity.

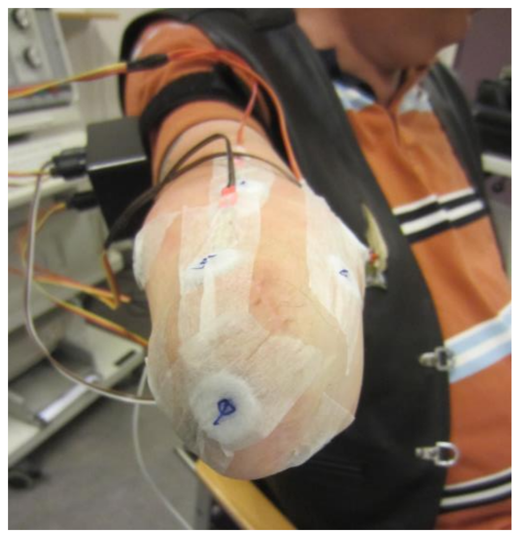

Figure 9 - Sample sensor placement

Procedure

Each participant was measured using both the Flex-Wrist and the Multiflex wrist with their corresponding hands. Each wrist was tested twice for a total of four test conditions; first all tests were performed with the wrists locked at the 0° neutral position to emulate static wrists (Otto Bock ® Flex Wrist static or OBS and Motion Control ® Multiflex wrist or MCS). Then for both wrists, users were allowed to choose the position of the wrist (OBF and MCF). Each participant was assigned a random condition order, with the only restriction that the conditions be paired by wrist type (i.e. if the first condition randomly assigned was OBF, then the second condition automatically became OBS. Then the third condition was randomly chosen).

Participants performed one baseline measurement and six tasks. All measurements were repeated four times. The measurements always started with a baseline (Table 4), in order to establish the pressure while not performing a task. Then six activities of daily life (ADL) (Table 4) were performed in a random order. This was repeated for each condition. The order of condition is shown in Table 5.

Table 4 - Description of ADL tasks.

Task	Task description
Baseline - Standing	Participants were asked to stand straight with their hands by their side with the prosthetic hand closed. Then they were asked to open and close their hand. Next they placed their arm in front of them at a 90° angle to their torso, and once again opened and closed their hand. Finally they raised their arm straight up at 180° from their torso and opened and closed their hand. They then returned to the initial position.
Turning door handle - One-handed - Standing	A door-handle, mounted on a wooden post at 24 cm high, was fixed at the corner of a foldable crate. Participants were standing with their heels at a taped line at 35 cm from the edge of the table. The door handle was placed right in front of the participants' affected arm at 25 cm from the edge of the table. Participants turned the door handle down and up with their prosthetic hand.
Lifting crate - Two-handed - Standing	Participants were standing with their heels at a taped line at 35 cm from the edge of the table. They lifted a foldable crate of 37 x 25 x 19 cm (length x width x height) and 1,54 kg with both hands from the affected to the unaffected side, over a box of 29 x 29 x 17 cm (length x width x height) that was located in front of the participant, at the edge of the table. Tape signs at 20 cm left and right of the box indicated start and end position of the crate. The crate had to be placed beyond these signs.

Closing and opening zip - Two-handed - Standing	Participants were fitted with a custom-made jacket with a zip (22 cm long) at the front, starting at hip height. Participants grabbed the zipper with their prosthetic hand and close and open the zip, while supporting the jacket with their unaffected hand. Participants were allowed to pass the zipper with their unaffected hand to their prosthetic hand when grabbing it.
Lifting object - One-handed - Seated	A box of 29 x 29 x 17 cm (length x width x height) was placed in front of the participant, at the edge of the table. A wooden cylinder (7 cm diameter, 14 cm high) was placed 14 cm next to the wooden box and 14 cm from the edge of the table, at the affected side of the participant. The participant grabbed the cylinder (power grip) and placed it on the top of the box, at the marked middle point.
Stirring into cup - One-handed - Seated	A cup filled with water was placed in front of the participant at 25 cm from the edge of the table. Participants picked up the spoon with their prosthetic hand, stirred 5 times, and released the spoon. Participants were allowed to pass the spoon with their unaffected hand to their prosthetic hand when grabbing it.
Handling cutlery - Two-handed - Seated	A piece of clay (5 cm long) was placed in front of the participant at 25 cm from the edge of the table. Participants held a fork in their prosthetic hand and a knife in their unaffected hand. From this starting position, participants fixated the clay with the fork, and cut it with the knife.

Table 5 - Test condition orders

Participant	Condition 1	Condition 2	Condition 3	Condition 4
P1	MCS	MCF	OBS	OBF
P2	OBS	OBF	MCF	MCS
P3	MCF	MCS	OBF	OBS
P4	OBF	OBS	MCS	MCF
P5	OBS	OBF	MCS	MCF
P6	MCS	MCF	OBF	OBS
P7	MCF	MCS	OBS	OBF
P8	OBF	OBS	MCF	MCS

Abbreviations: OBS – Otto Bock Static Condition, OBF – Otto Bock Flexible Condition, MCS – Motion Control Static Condition, MCF – Motion Control Flexible Condition.

Analysis

Each repetition was recorded as a separate data set. To facilitate analysis, each data set was resampled in MATLAB® R2012B to include 100 data points. To ensure the validity of the results produced by this set-up, each set of four repetitions was correlated to account for sensor inconsistency. All repetitions found to be uncorrelated were ignored for analysis, as they proved to be overly noisy.

As participants do not perform tasks in the same exact manner between repetitions, related pressure peaks appear at different points in the data. Therefore it is unreasonable to simply average all four repetitions. Instead, pressure peaks were identified, and those were averaged in both height and time of occurrence. MATLAB® R2012B was used to identify the maximum peaks in the data. The peaks were matched by location of occurrence and the maximum of the four repetitions was used.

Correlations and general linear models were done in SPSS (version 22, 2014). All analysis code can be found in Appendix 2. Repeated measures ANOVA was also performed with Manufacturer (OB vs. MC), Wrist type (Static vs. Flexible), and Task (Baseline, Door Handle, Crate Lift, Zipper, Object Lift, Cup Stirring, Cutlery Handling) were within-subject factors and Stump length (Short vs. Long) was a between-subject factor. The significance level was set at 0.05.

Results

To ensure the validity of the results produced by this set-up, first the four repetitions for each task were correlated. It was found that 90.3% of all repetitions had a significant correlation ($p<0.05$). The remaining repetitions were removed from the dataset. It can be easily seen that the setup provided reproducible results by comparing the four repetitions of a random selection of measurements (Figure 10).

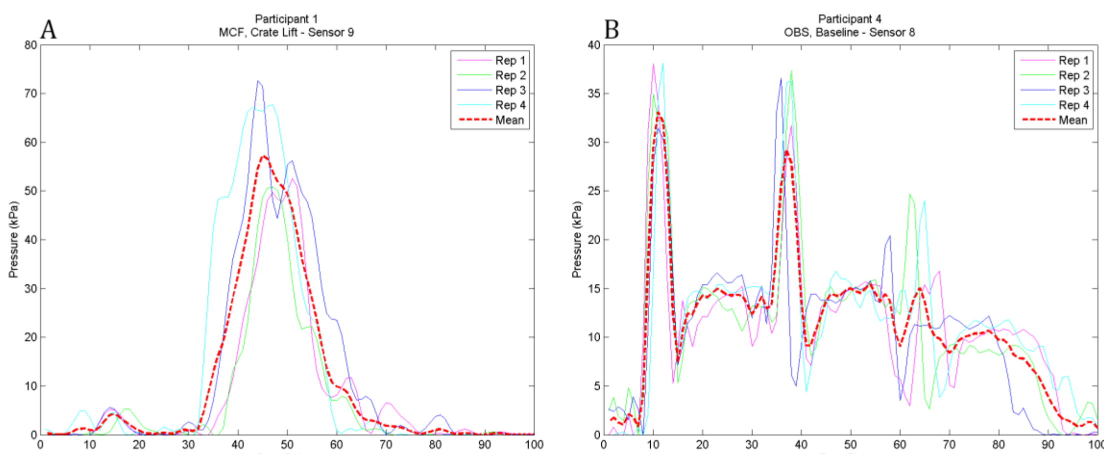

Figure 10 - - (A) Plot of resampled four repetitions for data recorded by sensor 9 during the Crate Lift task for participant 1. (B) Plot of resampled four repetitions for data recorded by sensor 8 during the Baseline for participant 4. It can be seen that all four repetitions form the same pattern. These plots are representative of the 90.3% of correlated data.

Once the peaks were identified and averaged, they were checked against the resampled repetitions to ensure correct peak identification. Figure 11 shows a representative sample of the identified averaged peaks compared to the peaks identified in each repetition.

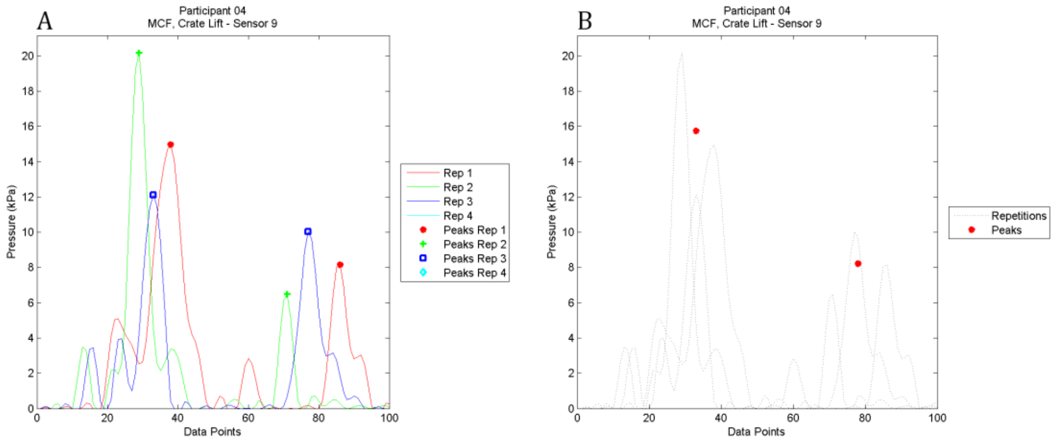

Figure 11 - – (A) Data from four repetitions with their respective identified peaks. The peaks were then averaged based on location and height. (B) shows the peaks obtained from this average.

Filled contour plots were made to better understand the pressure distribution inside the socket. A grid was made according to the location of the sensor on the arm (Table 3). The peaks were averaged across all repetitions for each hand, task, and sensor. Figure 12 sample pressure distribution plots. The highest areas of pressure somewhat varied between participants in the long group. The short group showed a similar distribution within the group. The task type had no effect on the pressure distribution. Overall, the static and flexible conditions had similar pressure distributions, and the elbow/ulnar and elbow/anterior areas usually had the highest pressure.

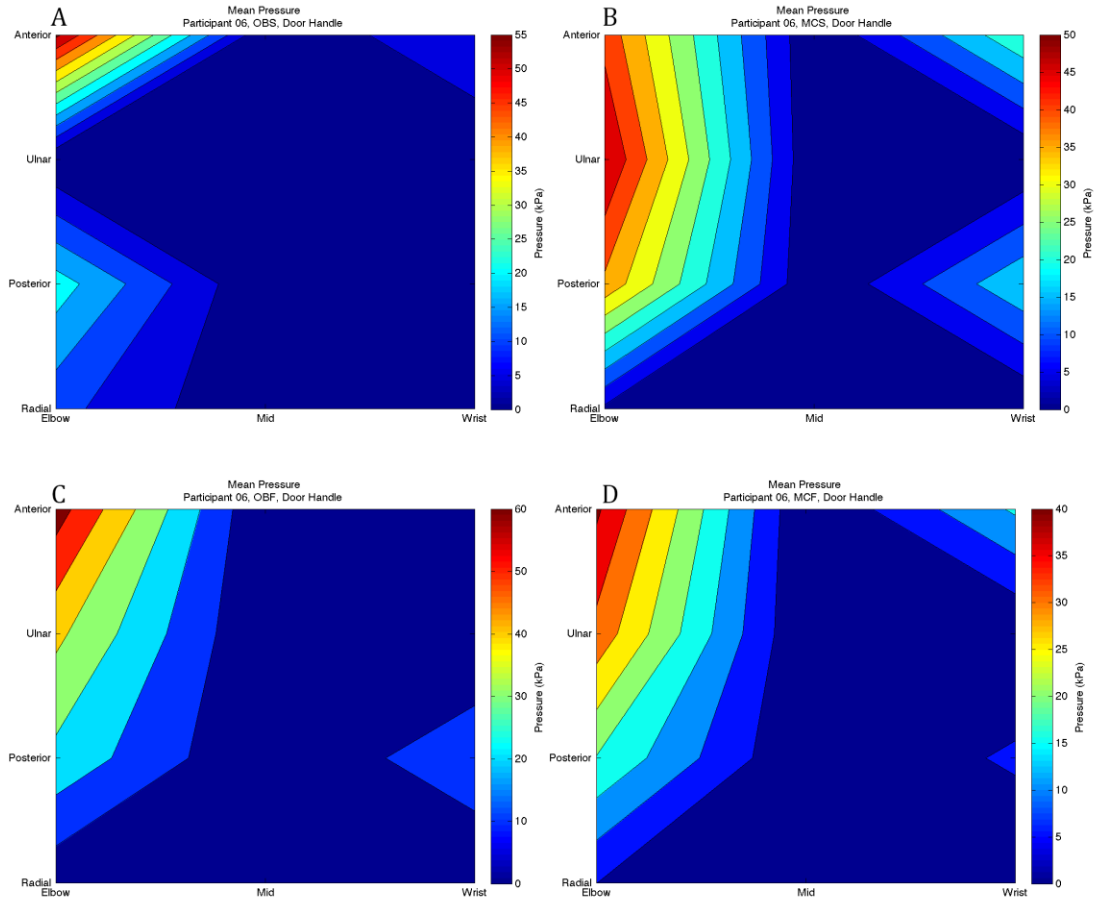

Figure 12 - Plot of pressure distribution during the Door Handle task for participant 6 for (A) OBS (B) MCS (C) OBF (D) MCF. It is clear that the type of wrist made little difference to the pressure distribution scheme.

From the repeated measure ANOVA, it was found that:

Manufacturer: $F(1, 6) = 9.38$; $p<0.05$ (OB = 40.81 kPa [32.49-49.13 kPa], MC = 46.62 kPa, with the confidence interval (CI) [35.06-58.17 kPa]). OB had a lower maximum pressure than MC.

Static and flexible conditions: $F(1, 6) = 24.24$; $p<0.005$ (Static = 42.32 kPa, with CI [32.95-51.68 kPa], Flexible = 45.11 kPa, with CI [34.85-55.37 kPa]). The static condition had a lower maximum pressure than the flexible.

Tasks: $F(6, 36) = 11.48$; $p<0.000$ (Baseline = 44.02 kPa, with CI [31.26-56.79 kPa], Door Handle = 41.73 kPa, with CI [27.15-56.31 kPa], Crate Lift = 49.85 kPa, with CI [38.21-61.50 kPa], Zipper = 61.34 kPa, with CI [48.38 kPa], Object Lift = 45.71 kPa, with CI [32.02-59.40 kPa], Cup Stirring = 31.11 kPa, with CI [22.71-39.50 kPa], Cutlery Handling = 32.24 kPa, with CI [23.73-40.75 kPa]). Zipper caused the highest maximum pressure, while Cup Stirring and Cutlery Handling cause similarly low pressure on the stump. Surprisingly, the Baseline was not appreciably different from the other tasks.

When comparing manufacturers between participant groups, OB and MC had similar pressure, however MC was much higher in the short group ($p<0.05$) (Figure 13 A). Similarly, no difference was seen in the static and flexible conditions in the long group, but the flexible was higher than static in the short group ($p<0.01$) (Figure 13 B). In general, tasks had a lower pressure in the long group than in the short group ($p<0.00$), with the exception of the baseline. Furthermore, although the pressure was higher in the short group, the same pattern is observed in both groups (Figure 13 C). Finally, the static and flexible conditions had different behaviors between manufacturers. While only a small difference was seen between manufacturers for the flexible condition, OB had a much lower pressure than MC in the static condition ($p<0.05$) (Figure 13 D). The reason behind these differences is not immediately obvious.

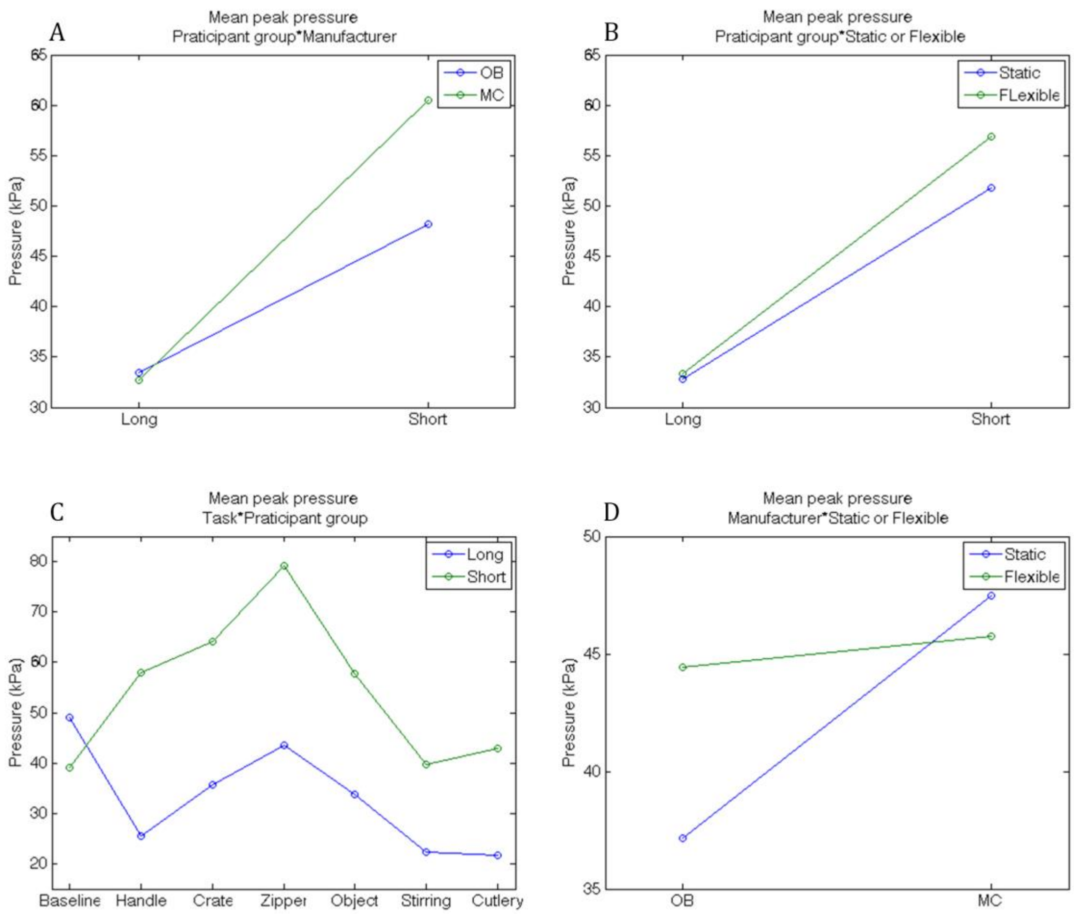

Figure 13 – (A) Interaction between manufacturer and participant group. The short long group shows no difference in pressure between manufacturers, while MC in much higher than OB in the short one. (B) Interaction between participant group and static or flexible conditions. Again, no difference is seen in the long group, while the flexible condition has a higher pressure in the short one. (C) Interaction between task and participant group. It can be seen that for all tasks besides the baseline, the short group has a higher pressure than the long one. (D) Interaction between manufacturer and static or flexible condition. The flexible condition shows little difference between manufacturers, while OB is much lower than MC in the static condition.

Experiment Discussion

When comparing the four different wrists used in this experiment, differences in pressure distribution, and mean peak pressure in the socket were compared. Differences in pressure distribution were only noticeable between participants of the long group, with no differences within each participant. This suggests that the distribution is affected mostly by how well the socket fits, as well as the morphology of the stump (i.e. areas with less tissue over the bone) in the long group. In the short group, however, as there is a smaller surface area where the pressure is distributed, these differences are much less prevalent. The highest pressure was usually seen in the elbow/ulnar and elbow/anterior areas, which is consistent with the mechanical load on the stump. Since most of the load is at the end of the stump (the hand), a torque is created and most of the force is applied in the elbow/ulnar area of the stump.

There were many behaviors that were surprising. It was observed that hand opening and closing causes higher pressure due to muscle activation during all tasks. Furthermore, as no external loads are applied during the Baseline, it was expected that it would have a lower pressure than other tasks. However it was seen that the baseline had a higher pressure than the Cutlery Handling and Cup Stirring, and similar pressure to the Object Lift. This is probably due to the weight of the hand being loaded on the utensils used in the Cutlery Handling and Cup Stirring, and thus the load on the stump was reduced. As for the Object Lift, the object was light compared to the hand, and so there was not a significant increase in the load.

Furthermore, OB had a lower pressure than MC, however this was shown to be due mostly to the static condition and the short group. The MC hand and wrist was 112 grams heavier and the center of mass was 0.85 cm farther from the arm than OB. These two factors caused this difference in the short group. The long group showed no difference due to the larger area where the pressure is distributed and thus the effects of the higher load were minimized.

It was also seen that in the long group, no difference was found between static and flexible conditions, and only a small difference (higher flexible) in the short group. It is interesting to note that with the Multiflex wrist by Motion Control ®, participants mostly chose to lock the wrist at 30° flexion rather than leaving it in the completely flexible position. This indicates that wrists that can be locked in flexion or extension positions offer no advantages over static wrists when it comes to pressure in the socket. However, further experiments need to be conducted to evaluate the advantages of an unlocked flexible wrist compared to a locked static wrist that allows for deviation.

Conclusion

In this project, a novel system for investigating the pressure inside upper limb prosthetic sockets was designed. Such pressure has never been investigated in literature before, which proved to be challenging as no information existed. Nonetheless, a systematic approach allowed this difficulty to be overcome and yielded a successful system that uses a Force Sensing Resistors connected to an Arduino. Whit this system, the different pressures caused by two different prosthetic wrists was measured.

Results showed no statistical difference in the pressure distribution between conditions. Furthermore, no appreciable difference was seen between static and flexible conditions in pressure, however MC had higher pressure than OB for the short group. There is no apparent reason for these difference therefore more investigation is needed.

When it comes to pressure in the socket, however, it is safe to conclude that wrists with additional degrees of freedom offer no advantages over wrists that can only rotate. Therefore, prosthetic users should choose wrists based on their preference and conformability with the system rather then the degrees of freedom of the wrist.

Recommendations

There are many areas where improvements can be made in this project, however the main concern that needs to be addressed by a future study is the 16.58% error between the measured and actual force. This was an artifact of the sensor used, and can only be mitigated by choosing a different sensor. Another drawback of this sensor is the inability to measure shear stress. It seems intuitive that most activities would not produce much shear stress, as the loads are mostly radial. Nonetheless it would be interesting to investigate such stress and correlate their location and intensity with skin problems.

Furthermore, this system was quite low resolution. Each sensor measured pressure in an area of 1.27 cm^2, and only a maximum of twelve sensors could be placed. Future studies should further investigate the woven force sensor, in order to decrease individual sensor size and increase the total area measured at the same time.

Finally, although most of the time the USB cable did not hinder the mobility of the participants, a simple modification to the design can make it wireless, provided Arduinos are still used. There are many Wi-Fi, Bluetooth and other radio add-ons readily available on the market that are small and light enough to fit in the control box, but these were not used due to minimize cost.

As for the study design, it would be interesting to repeat the experiment with a broader selection of tasks, as well as a greater number of participants. Tasks specifically designed where flexion or extension of the wrist would be beneficial, as they would increase the differences between flexible and static wrists. A greater number of participants would give more reliability to the results and allow more certain conclusions on the possible benefits of flexible wrists. Finally, unfortunately many sensors broke between measurements, and as a result, participants 1-4 were measure with 10 sensors while participants 5-8 were measure with only 6 sensors. It would have been better to have the same amount of sensors for all participants

References

1. Bertels, T., Schmalz, T., & Ludwigs, E. (2009). Objectifying the Functional Advantages of Prosthetic Wrist Flexion. *Journal of Prosthetics & Orthotics* , 74-78.
2. Carey, S. L., Highsmith, J. M., Maitland, E. M., & Dubey, R. V. (2008). Compensatory movements of transradial prosthesis users during common tasks. *Clinical Biomechanics* , 1128-1135.
3. Convery, P., & Buis, W. P. (1998). Conventional patellar-tendon-bearing (PTB) socket/stump interface dynamic pressure distributions recorded during the prosthetic stance phase of gait of a trans-tibial amputee. . *Prosthetics and Orthotics International* , 193-198.
4. Ephraim, P. L., Wegener, S. T., MacKenzie, E. J., Dillingham, T. R., & Pezzin, L. E. (2005). Phantom Pain, Residual Limb Pain, and Back Pain in Amputees: Results of a National Survey. *Archives of Physical Medicine and Rehabilitation* , 1910-1919.
5. Jones, D. H., Kilgour, R. D., & Comtois, A. S. (2007). Test-Retest Reliability of Pressure Pain Threshold Measurements of the Upper Limb and Torso in Young Healthy Women. *The Journal of Pain* , 650-656.
6. Metzger, A., Dromerick, A., Holley, R., & Lum, P. (2012). Characterization of compensatory trunk movements during prosthetic upper limb reaching tasks. *Arch Phys Med Rehabil* , 2029-2034.
7. Ostlie, K., Frankiln, R., Skjedal, O., & Skrondal, A. (2011). Assessing physical function in adult acquired major upper-limb amputees by combining the Disabilities of the Arm, Shoulder and Hand (DASH) Outcome Questionnaire and clinical examination. *Archives of Physical Medicine and Rehabilitation* , 1636-1645.
8. Sanders, J. E., & Daly, C. H. (1993). Normal and shear stresses on a residual limb in a prosthetic socket during ambulation : Comparison of finite element results with experimental measurements. *Jornal of Rehabilitation Research* , 191-204.
9. Satomi, M., & Wilson, P. (2013). *Sensors - Woven Pressure Sensor Matrix.* Retrieved from KobaKant: http://www.kobakant.at/DIY/?p=4296
10. Zinck, A., Stavdahl, O., Biden, E., & Kyberd, P. (2012). Design of a compact, reconfigurable, prosthetic wrist. *Applied Bionics and Biomechanics* , 117-124.
11. Interlink Electronincs. (n.d.). *Interlink Electronics FSR® Force Sensing Resistors®.* Retrieved from Interlink Electronincs: www.interlinkelectronincs.com

Appendix

Appendix 1 – Data Collection

Clear all

```
Snd('Open');
clc;
clearvars;
clearvars -global;
commandwindow;
```

Specifies the COM port that the arduino board is connected to

```
if ismac; comPort='/dev/tty.usbmodemfa1421'; end;
if ispc; comPort='COM11'; end;
%connect MATLAB® R2012b to the sensor
if (~exist('serialFlag','var'))
    [fsr.s,serialFlag] = setupSerial(comPort);
end
```

Calibrate the sensors if no calibration file is present

```
if(exist('calibration.mat','file'))
    load('calibration.mat','m','weights','P');
else
    [P] = calibration(fsr);
end
```

Set Condition order

```
hands={'OBS', 'OBF', 'MCS', 'MCF'};
ADL={'Door Handle', 'Crate Lift', 'Zipper', 'Object Lift', 'Cup Stirring', 'Cutlery
Handling'};
```

Initializing variables

```
global OBS OBF MCS MCF;
beep = MakeBeep(450,0.2);
area=pi*0.00635^2; % Sensor area in meters squared
keyIsDown = 0;
if ismac; ESC=41; Quit=20; end;
if ispc; ESC=27; Quit=81; end;
i=1;
k=1;
```

Acquire data and plot

```
for h=1:length(hands)
    key=0;

    % Test if sensors are working
    while key~=ESC
        [keyIsDown,~,keyCode] = KbCheck();
        keyCode=[keyCode 1];
        key=find(keyCode,1);
        [voltage]=readFSR(fsr); % Read data from FSR

        for l=1:12
            mass(l)=polyval(P(l,:), voltage(l))/1000-polyval(P(l,:), 0)/1000; % In kg
        end

        pressure(i,:)=(mass*9.81)/(area*1000); % In kPa

        % Update the readouts
        dispstat(sprintf(['\n         Sensor Test\n' ...
            '         ___________________________________________________________\n\n' ...
            '         Sensor  1     Sensor  2     Sensor  3     Sensor  4 \n
%7.2f       %7.2f       %7.2f       %7.2f  (kPa) \n \n' ...
            '         Sensor  5     Sensor  6     Sensor  7     Sensor  8 \n
%7.2f       %7.2f       %7.2f       %7.2f  (kPa) \n \n' ...
            '         Sensor  9     Sensor 10     Sensor 11     Sensor 12 \n
%7.2f       %7.2f       %7.2f       %7.2f  (kPa) \n' ...
            '         ___________________________________________________________\n\n
Press ESC to end sensor testing.'] ...
            ,pressure(i,1),pressure(i,2),pressure(i,3),pressure(i,4) ...
            ,pressure(i,5),pressure(i,6), pressure(i,7),pressure(i,8) ...
            ,pressure(i,9),pressure(i,10),pressure(i,11),pressure(i,12) ));

        pause(.1);
        i=i+1;

        if key==Quit
            break;
        end
    end
    clearvars pressure;

    seq = 2:length(ADL)+1;
    seq = seq(randperm(length(seq)));

    for p=1:length(ADL)
        ADLs(1,p)=ADL(seq(p)-1);
    end

    seq = [1, seq];
    stages=['Baseline' ADLs 'End'];
```

```
    for i=1:length(stages)
        if i<length(stages)-1
            printstages{i}=[stages{i} ', '];
        else
            printstages{i}=stages{i};
        end
    end

    stagetimes=zeros(1,length(stages)-1);
    timewait=0;

    dispstat(sprintf(['\n         Exectue tasks using ' hands{h} '.\n\n         Please
press space to start tasks in the following order:\n\n         '  printstages{1,1:end-1}
'.']));

    KbWait([], 2);
    key=0;
    all=[];
    tic;

    for j=1:length(stages)-1
        for n=1:4
            i=1;
            Snd('Play',beep);
            key=0;

            while key~=ESC
                [keyIsDown,~,keyCode] = KbCheck();
                keyCode=[keyCode 1];
                key=find(keyCode,1);

                [voltage]=readFSR(fsr); % Read data from FSR
                rawdata(i,:)=voltage;

                for l=1:12
                    mass(l)=polyval(P(l,:), voltage(l))/1000-polyval(P(l,:), 0)/1000; %
In kg
                end

                pressure(i,:)=(mass*9.81)/(area*1000); % In kPa
                % Update the readouts

                time(k)=toc;

                if j>1
                    currtime(i)=time(k)-timewait;
                else
                    currtime(i)=time(k);
                end

                dispstat(sprintf(['\n         ' stages{j} ' (' int2str(n) '\\4). Task '
```

```
int2str(j) ' of ' int2str(length(stages)-1) '. Next task is ' stages{j+1} '.'...
                    '\n         Elapsed Time - %2.2d:%2.2d:%2.2d \n\n' ...
                    '
__________________________________________________________\n\n' ...
                    '         Sensor  1     Sensor  2     Sensor  3     Sensor  4 \n
%7.2f       %7.2f       %7.2f       %7.2f  (kPa) \n \n' ...
                    '         Sensor  5     Sensor  6     Sensor  7     Sensor  8 \n
%7.2f       %7.2f       %7.2f       %7.2f  (kPa) \n \n' ...
                    '         Sensor  9     Sensor 10     Sensor 11     Sensor 12 \n
%7.2f       %7.2f       %7.2f       %7.2f  (kPa) \n' ...
                    '
__________________________________________________________\n' ...
                    '\n\n         Total Elapsed Time - %2.2d:%2.2d:%2.2d \n\n
Press ESC to end section or Q to quit.'] ...

,floor(currtime(i)/3600),floor(rem(currtime(i)/60,60)),floor(rem(currtime(i),60)) ...
                    ,pressure(i,1),pressure(i,2),pressure(i,3),pressure(i,4) ...
                    ,pressure(i,5),pressure(i,6), pressure(i,7),pressure(i,8) ...
                    ,pressure(i,9),pressure(i,10),pressure(i,11),pressure(i,12) ...
                    ,floor(time(k)/3600),floor(rem(time(k)/60,60)),floor(rem(time(k),60))
));

                pause(.1);
                i=i+1;
                k=k+1;

                if key==Quit
                    break;
                end
            end

            if key==Quit
                break;
            end

            stagetimes(n)=time(k-1);

            eval([hands{h} '.pressure_' int2str(seq(j)) '_' int2str(n) '=pressure;']);
            eval([hands{h} '.rawdata_' int2str(seq(j)) '_' int2str(n) '=rawdata;']);
            eval([hands{h} '.stagetime_' int2str(seq(j)) '_' int2str(n) '=currtime'';']);
            clearvars marker temp pressure rawdata currtime;

            if n<4
                dispstat(sprintf(['\n         Please press space to perform repetition '
int2str(n+1) ' for ' stages{j} '.']));
                KbWait([], 2);
            end
            timewait=toc;
        end


        if j<length(stages)-1
            if key==Quit
```

```
                break;
            end
            dispstat(sprintf(['\n         Please press space to continue to ' stages{j+1}
'.']));
            KbWait([], 2);
        end

        if key==Quit
            break;
        end
    end

    eval([hands{h} '.time=time'';']);
    eval([hands{h} '.stages=stages;']);

    clear time stages

    if key==Quit
        break;
    end
end
```

Format and saves data

```
matfilename=sprintf([pwd '/Data/data_%s.mat'],datestr(now,'yyyy-mm-dd_HH-MM-SS'));
save(matfilename,'OBS', 'OBF', 'MCS', 'MCF', 'hands');

closeSerial;
Snd('Close');
clearvars;
clearvars -global;
```

Published with MATLAB® R2012B

Appendix 2 – Data Analysis

Repetition Correlation

```
clc
clear all
hands={'OBS', 'MCS', 'OBF', 'MCF'};
part={'01','02','03','04','05','06','07','08'};
reps=[1 2 3 4];
badreplist = fopen([pwd '/Cleaned/badreplist.txt'],'w');
badreppercent = fopen([pwd '/Cleaned/badreppercent.txt'],'w');

for p=1:8
    cntrtotal=0;
    cntrbad=0;
    load([pwd '/Resampled/' part{p} '/' part{p} '_rsmp.mat']);

    for i=1:length(hands)
        for j=1:7
            for k=1:10
                eval([ 'temp=[' hands{i} '.rawdata_' int2str(j) '_1(:,k) ' hands{i}
'.rawdata_' int2str(j) '_2(:,k) ' ...
                    hands{i} '.rawdata_' int2str(j) '_3(:,k) ' hands{i} '.rawdata_'
int2str(j) '_4(:,k)];' ]);

                [~,tempp]=corr(temp);
                tempp(triu(tempp,1)==0)=1;
                [tempi,tempj] = find(tempp<0.05);
                corrReps=unique([tempi,tempj]);
                badreps=setdiff(reps,corrReps);

                for l=1:length(badreps)
                    if any(temp(:,l))
                    fprintf(badreplist,[part{p} '_' hands{i} '_' int2str(j) '_'
int2str(k) '_' int2str(badreps(l)) '\n']); cntrbad=cntrbad+1;
                    end

                    eval([ hands{i} '.rawdata_' int2str(j) '_' int2str(badreps(l))
'(:,k)=nan(100,1);' ]);
                end

                for m=1:4
                    cntrtotal=cntrtotal+1;
                end
            end
            clear temp*
        end
    end
```

```
    save([ pwd '/Cleaned/' part{p} '_clean.mat'],'OBS', 'OBF', 'MCS', 'MCF', 'hands');
    fprintf(badreppercent,[part{p} ' - %1.2f%% (%1.0f/%1.0f) p>0.05\n'],
(cntrbad/cntrtotal)*100,cntrbad,cntrtotal);
    fprintf([part{p} ' - %1.2f%% (%1.0f/%1.0f) p>0.05\n'],
(cntrbad/cntrtotal)*100,cntrbad,cntrtotal)
end
fclose(badreplist);
fclose(badreppercent);
```

Published with MATLAB® R2012B

Data Resampling

```
clc
close all
clear java

for p=1:8

    fprintf('Participant %0i\n',p);
    hands={'OBS', 'MCS', 'OBF', 'MCF'};
    color={'m', 'g', 'b', 'c'};
    part={'01','02','03','04','05','06','07','08'};
    tasks={'Baseline','Door Handle', 'Crate Lift', 'Zipper', 'Object Lift', 'Cup
Stirring', 'Cutlery Handling'};

    for i=1:length(hands)

        fprintf('     Hand %0i\n',i);
        load([pwd '/Original/' part{p} '.mat'], hands{i})

        for j=1:7

            fprintf('          Task %0i\n',j);

            for k=1:4

                eval([ 'temp=' hands{i} '.rawdata_' int2str(j) '_' int2str(k) '(:,1:10);'
]);
                curlength=length(temp);
                temp=resample(temp,100,curlength);
                temp(temp<0)=0;
                eval([ hands{i} '_.rawdata_' int2str(j) '_' int2str(k) '= temp;' ]);

                eval([ 'temptime=' hands{i} '.stagetime_' int2str(j) '_' int2str(k) ';'
]);
                temptime=resample(temptime,100,curlength);
                eval([ hands{i} '_.stagetime_' int2str(j) '_' int2str(k) '= temptime;'
]);
```

```
            for l=1:10
                figure(l);
                plot(temp(:,l),color{k});
                hold on;
            end

            clear curlength temp temptime
        end

        for l=1:10
            eval([ 'tempmean=nanmean([' hands{i} '_.rawdata_' int2str(j) '_1(:,l) '
hands{i} '_.rawdata_' int2str(j) '_2(:,l) ' hands{i} '_.rawdata_' int2str(j) '_3(:,l) '
hands{i} '_.rawdata_' int2str(j) '_4(:,l)' '],2);' ]);
            eval([ hands{i} '_.mean_' int2str(j) '_' int2str(l) '= tempmean;' ]);
            figure(l);
            plot(tempmean,'--r','LineWidth',2)
            title(sprintf([ 'Participant ' int2str(p) '\n' hands{i} ', ' tasks{j} ' -
Sensor ' int2str(l) ]))
            legend('Rep 1','Rep 2','Rep 3','Rep 4','Mean');
            xlabel('Data Points');
            ylabel('Pressure (kPa)');
            filename=[pwd '/Resampled/' part{p} '/Figures/Reps/reps_' hands{i} '_'
int2str(j) '_' int2str(l) '.png'];
            saveas(l,filename);
            close(l);
        end
    end

    eval([ 'clear ' hands{i} ]);
    eval([ hands{i} '=' hands{i} '_;' ]);
    eval([ 'clear ' hands{i} '_' ]);

    save([hands{i} '.mat'], [hands{i}]);

    eval([ 'clear ' hands{i} ]);
end

for i=1:4
    load([hands{i} '.mat']);
end

save([ pwd '/Resampled/' part{p} '/' part{p} '_rsmp.mat'],'OBS', 'OBF', 'MCS', 'MCF',
'hands');

for i=1:4
    delete([hands{i} '.mat']);
end

clear java
end
```

Published with MATLAB® R2012B

```
clear all
warning off;

hands={'OBS', 'MCS', 'OBF', 'MCF'};
part={'01','02','03','04','05','06','07','08'};
load([pwd '/calibration.mat'],'P');
P=mean(P);

fileID = fopen('badreplist.txt');
badreps = textscan(fileID,'%s');
fclose(fileID);

nopeaks = fopen([pwd '/Pks/nopeaks.txt'],'w');
nonesame = fopen([pwd '/Pks/nonesame.txt'],'w');
twosame = fopen([pwd '/Pks/twosame.txt'],'w');
twotwosame = fopen([pwd '/Pks/twotwosame.txt'],'w');
threesame = fopen([pwd '/Pks/threesame.txt'],'w');
foursame = fopen([pwd '/Pks/foursame.txt'],'w');
nocntr=1; nonecntr=1; twocntr=1; twotwocntr=1; threecntr=1; fourcntr=1;

for p=1:8

    load([pwd '/Cleaned/' part{p} '_clean.mat']);

    for i=1:length(hands)

        for j=1:7
            MPKS=zeros(100,10);

            for k=1:10
                for l=1:4
                    eval([ 'temp' int2str(l) '=' hands{i} '.rawdata_' int2str(j) '_'
int2str(l) '(:,k);' ]);
                end

                tempmean=nanmean(nanmean([temp1 temp2 temp3 temp4]));

                %finds peaks in data
                loccntr=[0 0 0 0];
                for l=1:4
                    eval([ 'badchk=''' part{p} '_' hands{i} '_' int2str(j) '_' int2str(k)
'_' int2str(l) ''';' ]);
                    if ~any(strcmp(badchk,badreps{1}))
                        loccntr(l)=1;
                        eval([ '[PKS' int2str(l) ',LOCS' int2str(l) ']=findpeaks(temp'
int2str(l) ',''MINPEAKHEIGHT'',2*tempmean,''MINPEAKDISTANCE'',10,''THRESHOLD'',0.5);' ]);
                        eval([ 'temppk=PKS' int2str(l) ';' ]);
                        eval([ 'tempemptycheck=nanmean(' hands{i} '.rawdata_' int2str(j)
```

```
'_' int2str(l) '(:,k));' ]);
                        if isempty(temppk) && any(tempemptycheck);

fprintf(nopeaks,[int2str(nocntr) ' - ' part{p} ' - ' hands{i} '_' int2str(j) '_'
int2str(k) ' - Rep ' int2str(l) '\n']); nocntr=nocntr+1; end;
                    end
                end
                loccntr=find(loccntr==1);

                if isempty(loccntr)
                    lengths=[];
                else
                    for l=1:length(loccntr)
                        if loccntr(l)
                            eval([ 'lengths(l)=length(LOCS' int2str(loccntr(l)) ');' ]);
                        end
                    end
                end
                sortlen=sort(lengths);
                tempreps=find(diff(sortlen)==0);

                %determines number of peaks to use for four reps
                if length(lengths)==4
                    if isempty(tempreps) %if all reps have different number of peaks,
take minimum
                        minlength=min(sortlen);
                        fprintf(nonesame,[part{p} ' - ' hands{i} '_' int2str(j) '_'
int2str(k) '\n']); nonecntr=nonecntr+1;
                    elseif length(tempreps)==1 %if all reps have one pair of same number
of peaks, take from pair
                        minlength=sortlen(tempreps);
                        fprintf(twosame,[part{p} ' - ' hands{i} '_' int2str(j) '_'
int2str(k) '\n']); twocntr=twocntr+1;
                    elseif length(tempreps)==2 && isempty(find(tempreps==2, 1)) %if all
reps have two pairs of same number of peaks, take from max pair
                        if min(sortlen)==0
                            minlength=max(sortlen);
                        else
                            minlength=min(sortlen);
                        end
                        fprintf(twotwosame,[part{p} ' - ' hands{i} '_' int2str(j) '_'
int2str(k) '\n']); twotwocntr=twotwocntr+1;
                    elseif length(tempreps)==2 && ~isempty(find(tempreps==2, 1)) %if all
reps have three of same number of peaks, take from triplet
                        minlength=sortlen(2);
                        fprintf(threesame,[part{p} ' - ' hands{i} '_' int2str(j) '_'
int2str(k) '\n']); threecntr=threecntr+1;
                    elseif length(tempreps)==3 %if all reps have the same number of
peaks, take from first
                        minlength=sortlen(1);
                        fprintf(foursame,[part{p} ' - ' hands{i} '_' int2str(j) '_'
int2str(k) '\n']); fourcntr=fourcntr+1;
```

```
                end
            elseif length(lengths)==3
                if isempty(tempreps) %if all reps have different number of peaks,
take minimum
```

```
                    minlength=min(sortlen);
                    fprintf(nonesame,[part{p} ' - ' hands{i} '_' int2str(j) '_'
int2str(k) '\n']); nonecntr=nonecntr+1;
                elseif length(tempreps)==1 %if all reps have one pair of same number
of peaks, take from pair
                    minlength=sortlen(tempreps);
                    fprintf(twosame,[part{p} ' - ' hands{i} '_' int2str(j) '_'
int2str(k) '\n']); twocntr=twocntr+1;
                else %if all reps have three of same number of peaks, take from
triplet
                    minlength=sortlen(1);
                    fprintf(threesame,[part{p} ' - ' hands{i} '_' int2str(j) '_'
int2str(k) '\n']); threecntr=threecntr+1;
                end
            elseif length(lengths)==2
                if isempty(tempreps) %if all reps have different number of peaks,
take minimum
                    minlength=min(sortlen);
                    fprintf(nonesame,[part{p} ' - ' hands{i} '_' int2str(j) '_'
int2str(k) '\n']); nonecntr=nonecntr+1;
                else
                    minlength=sortlen(tempreps);
                    fprintf(twosame,[part{p} ' - ' hands{i} '_' int2str(j) '_'
int2str(k) '\n']); twocntr=twocntr+1;
                end
            elseif length(lengths)==1
                minlength=sortlen(1);
            elseif isempty(lengths)
                minlength=0;
            end

            LOC=zeros(minlength,length(lengths)); PK=LOC;

            %choose which peaks to compare all to, based on number of peaks
            for l=1:length(loccntr)
                if loccntr(l)
                    eval([ 'temploc=LOCS' int2str(loccntr(l)) ';' ]);
                    if length(temploc)==minlength
                        comp=temploc;
                    end
                    clear temploc
                end
            end

            for l=1:length(loccntr)
                if loccntr(l)
```

```
                    eval([ 'templocs=LOCS' int2str(loccntr(l)) ';' ]);
                    if length(templocs)<minlength
                        templocs=padarray(templocs,minlength-
length(templocs),NaN,'post');
                        eval([ 'LOCS' int2str(loccntr(l)) '=templocs;' ]);
                        clear templocs
                    end
                end
            end

            for l=1:length(lengths)
                for m=1:minlength
                    eval([ 'loc=LOCS' int2str(loccntr(l)) ';' ]);
                    eval([ 'pk=PKS' int2str(loccntr(l)) ';' ]);
                    if ~isempty(loc)
                        [~, idx] = min(abs(loc - comp(m)));
                        LOC(m,l)=loc(idx);
                        PK(m,l)=pk(idx);
                        clear idx loc pk
                    else
                        LOC(m,l)=NaN;
                        PK(m,l)=NaN;
                    end
                end
            end

            %replace repeated peaks with NaN
            for l=1:length(lengths)
                if isempty(comp)
                    diffcomp=[];
                else
                    diffcomp(:,l)=comp;
                end
            end

            if ~isempty(LOC)
                diffmat=abs(LOC-diffcomp);
                LOC(diffmat>10)=NaN;
                PK(diffmat>10)=NaN;
                LOCMEAN=floor(nanmean(LOC,2));
                PKMEAN=nanmean(PK,2);
            end

            for l=1:minlength
                MPKS(LOCMEAN(l),k)=PKMEAN(l);
            end

            clearvars LOC* PK* temp* comp diffmat diffcomp k l minlength sortlen m
lengths loccntr -except PKS

        end
        MPKS(MPKS==0)=NaN;
        MPKSheight=nanmean(MPKS);
```

```
            for l=1:10
                MPKSnr(:,l)=length(find(~isnan(MPKS(:,l))==1));
            end

            eval([ 'PKS.' hands{i} '_' int2str(j) '=MPKS;' ]);
            eval([ 'PKS.' hands{i} '_' int2str(j) '_nomhgt=MPKSheight;' ]);
            eval([ 'PKS.' hands{i} '_' int2str(j) '_nrpks=MPKSnr;' ]);

            clear MPKS*

        end
    end
    save([ pwd '/Pks/' part{p} '_pks.mat'],'PKS');

end

fprintf(nopeaks,'\nRepetition percentage without peaks %1.2f\n',(nocntr/8960)*100);
fprintf(nonesame,'\nRepetition percentage without peaks in common
%1.2f\n',(nonecntr/2240)*100);
fprintf(twosame,'\nRepetition percentage with two peaks in common
%1.2f\n',(twocntr/2240)*100);
fprintf(twotwosame,'\nRepetition percentage with two pair of peaks in common
%1.2f\n',(twotwocntr/2240)*100);
fprintf(threesame,'\nRepetition percentage with three peaks in common
%1.2f\n',(threecntr/2240)*100);
fprintf(foursame,'\nRepetition percentage with four peaks in common
%1.2f\n',(threecntr/2240)*100);
fclose(nopeaks); fclose(nonesame); fclose(twosame); fclose(twotwosame);
fclose(threesame); fclose(foursame);

warning on;
```

Published with MATLAB® R2012B