Sensory Mapping

BME 300/200
Department of Biomedical Engineering
University of Wisconsin-Madison
December 12, 2008

Colleen Farrell, BSAC
Adam Pala, BWIG
Jeremy Schaefer, Team Leader
Steve Wyche, Communicator

Client
Dr. Miroslav Backonja, M.D
Department of Neurology, UW-Hospital

Advisor
Professor Mitch Tyler, Professor
Department of Biomedical Engineering

Table of Contents

Problem Statement

Dr. Miroslav Backonja, a neurologist who works in pain medicine at UW Hospital, has expressed the need for a more accurate method to measure the surface area of cutaneous sensory abnormalities. Currently, tracing paper is used to trace the affected area and a planimeter is used to measure surface area. Dr. Backonja is looking to be able to measure surface area on contoured regions of the body in an accurate and repeatable manner.

Background Information

Neuropathic pain presents as various forms of abnormal cutaneous sensations. These abnormalities can present as shooting pains, tingling, burning or numbness on the skin's surface (Pain 1). Neuropathic pain can be a result of many different illnesses or the treatments for these diseases. It can be painful diabetic neuropathy (PDN) or painful HIV-associated neuropathy (HIV-DSP). It can also be a result of illnesses such as multiple sclerosis, cancer, and spinal cord injuries (Neuropathic 1). The pain intensity ranges from that of a sunburn to intense shooting pains. Due to the fact that neuropathic pain tends to be caused by other illnesses, the pain can prove debilitating when it occurs concurrently with the symptoms from the disease the patient already suffers from.

In order for the clinician to help the patient measure the pain and other symptoms they must monitor the changes in a patient's neuropathy to track the progress or effectiveness of various treatments. To treat milder pain the patient may be given non-steroidal anti-inflammatory drugs while more severe pain may be treated with stronger painkillers such as morphine (Neuropathic 2). Management of the disease which is

causing the neuropathic pain, such as diabetes, can also help to alleviate some of the symptoms.

Design Motivation

The current method that clinics use to record the area of pain uses tracing paper to make a 2D representation of the 3D surface. The first step of this process is to map out the sensory anomalies on the patient's skin by using various brushes and tools to stimulate the surface. Based on the reaction of the patient to the stimulation, the areas of each different sensation are marked and then these markings can be traced. The tracing paper is wrapped on the surface as best as possible and each region is traced separately. Once there is a 2D representation of a region on the tracing paper, a planimeter is used to find the surface area.

There are a number of problems with this method of measuring the surface area. First, the tracing paper cannot be properly placed on the skin which leads to an inaccurate representation of the region which is being measured. These inaccuracies become very large when dealing with contoured surfaces of the body such as an armpit or face. The inability of the clinician to obtain an accurate measurement prevents the results from being reproducible. Generally, this outdated method does not take advantage of technology which could be used to obtain much more accurate results. In addition to improving the results, better technology could allow for more organization, faster results, and data sharing between physicians.

Performance Requirements

Having collaborated with the client, Dr. Backonja, the following performance requirements are addressed in the final design prototype (See Appendix E regarding the Product Design Specifications for more technical and logistical details).

1. ***Function***. The device is intended for clinical usage. Therefore, it should be portable and easy to use. Given that the client deals with patients having sensory abnormalities (i.e. hypersensitivity), the device should be relatively small, while minimizing contact with the skin, in order to avoid inducing excessive harm on the patient during clinical testing and application (some contact with the skin is necessary, however). As a corollary, the procedure should be systematized in order to reduce clinician error as much as possible.

2. ***Data Management and User Interaction with the Device***. The data which is collected by the device should be stored in a simple, effective manner. This is accomplished using computer software, which interfaces with the device via USB, in order to collect, store, and process the data, using OptiTrack's Point Cloud software, Microsoft Excel, and MATLAB (respectively). These steps can be performed by the clinician by following a simple protocol which relates these three programs (this is all done on the computer to which the device is connected).

3. ***Accuracy and Reliability***. The goal of the device is to be able to compute the surface area of a given region of interest on the body. The client has established an acceptable error rate of five- to ten-percent error for the calculation of the region of interest's surface area (with respect to the current tracing-paper-and-plenimeter method of calculating surface area). The device should also produce consistent, reproducible results for a given region of interest, given the clinical nature of its application. Optionally, the region of interest may be digitally reconstructed via software in order to provide the clinician insight into a given trial by visualizing the region of interest (this may in turn facilitate in refining the data collection process, in an attempt to reduce clinician error).

Competition

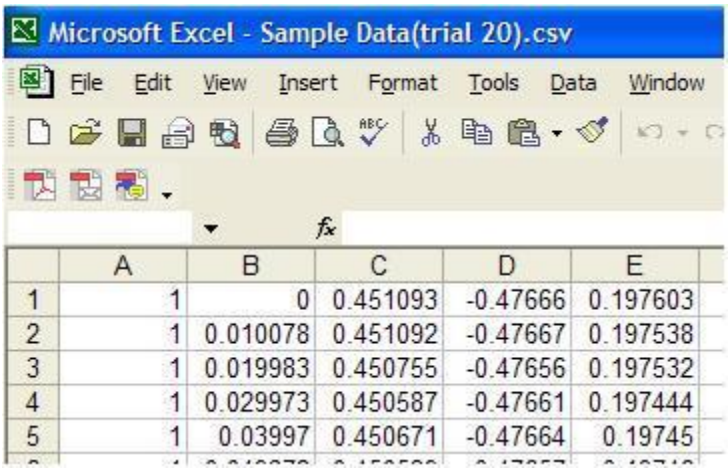
There doesn't appear to be a commercial product which deals directly with the problem of finding the surface area of a patient's skin. There is, however, a software program called BurnCase 3D which deals with burn victims (BurnCase 1). The software allows the doctor to trace on a 3D body the areas which are burned and differentiate between the severity of the burns. It then calculates the percentage of the body which is covered by each type of burn. The problem with this, however, the program doesn't provide any way in which to alter the given model. Without being able to change the body type or characteristics of the patient, tracing the affected regions on this model will not give very effective results. The only other programs which make 3D models from 2D pictures are very basic and don't involve dimensions.

Data Analysis

The data analysis portion of the proposed final design uses triangulation as an approximation method for the given region of interest being tested by the device. The reason for employing this approximation method is that, given the five- to ten-percent accuracy constraint of the surface area calculation (as stipulated by the client), this method will reasonably approximate the region of interest using triangles, provided that it is adequately represented by the collected data points (which form the vertices of these surface-approximating triangles), while allowing a relatively simple computation of the region of interest's surface area (which is determined by the collective area of these constituent triangles, as opposed to more complicated interpolative shapes, whose areas may be more difficult to compute).

When the data points are collected using the three OptiTrack Flex V100 infrared cameras, which interface via USB with a computer and are managed using the manufacturer-provided OptiTrack Point Cloud software, the raw data of a given trial is outputted in a comma-separated file (.csv format) containing five columns (listed in the respective order A to E): the tracked object (given proper calibration, only one object, the LED tip of the stylus, is tracked); the time (relative to the initial time when the capturing mode is initiated, with the last row of output corresponding to the time that the capturing mode is terminated); the *x*-coordinate (in meters); the *y*-coordinate (in meters); the *z*-coordinate (in meters). See Figure 1 (below) for an example of such an output.

Using this raw data, MATLAB is used to process and analyze it, with the goal of interpolating the region of interest from the collected data points and calculating its surface area. First, Columns C, D, and E of the .csv data file (corresponding to the three



	A	B	C	D	E
1	1	0	0.451093	-0.47666	0.197603
2	1	0.010078	0.451092	-0.47667	0.197538
3	1	0.019983	0.450755	-0.47656	0.197532
4	1	0.029973	0.450587	-0.47661	0.197444
5	1	0.03997	0.450671	-0.47664	0.19745

Figure 1: An example of output data from the OptiTrack Point Cloud software (as viewed in Microsoft Excel). Column A represents the tracked point; Column B represents the time of capture (relative to the initiation of the capturing mode); Column C represents the *x*-coordinate (in meters); Column D represents the *y*-coordinate (in meters); Column E represents the *z*-coordinate (in meters).

spatial dimensions of each point) are extracted into a data matrix and scaled by a factor of 100 (in order to convert from meters to centimeters, in all three spatial dimensions). With the understanding that the *y*- and *z*-coordinates constitute the two-dimensional plane which is perpendicular to the view of the cameras (thus the remaining *x*-coordinate is the "depth" dimension of the region of interest), Delaunay triangulation, a mathematical algorithm, is used to determine the constituent triangles which interpolate the region of interest (Delaunay Triangulation 1). This is done as follows: the *delaunay* function of MATLAB, which implements the Delaunay triangulation mathematical algorithm using MATLAB code, takes the *y*- and *z*-coordinates as its two arguments (as previously mentioned, these two dimensions constitute a two-dimensional plane which is perpendicular to the view of the cameras) and finds each planar point's nearest neighbor. This information is then outputted as a matrix which indexes these nearest-neighbor points (vertices of triangles). See Figure 2 (below) for an example of such an output in MATLAB.

Figure 2: An example of MATLAB output resulting from the application of the *delaunay* function. Having received two column vectors from the scaled-data matrix (containing the *y*- and *z*-coordinates of this particular point-collection trial) as its two arguments, the *delaunay* function produces a matrix of vertices which indexes all of the nearest-neighbor points of the initial column-vector arguments in the *yz*-plane (i.e. each row of output in the figure represents a given point's nearest-neighbor index in the *yz*-plane with respect to the scaled-data matrix containing the three-dimensional data points; as an example, the first row of output indicates that MATLAB will use the 270th, 753rd,and the 228th points of the scaled-data matrix to form a triangle). This particular trial resulted in a data matrix containing 1566 total rows, corresponding to 1566 collected points (all of which are used to define vertices of tbe constituent triangles which represent the region of interest).

Once these nearest neighbors are determined, the depth dimension (the *x*-coordinate) is applied to form the constituent triangles which approximate the three-dimensional region of interest. With the region of interest now being approximated by these constituent triangles (in three dimensions), determining the surface area of the region of interest merely requires a computation of the surface areas of each constituent triangle, thereby adding each contribution to obtain the total surface area of the region of interest; this is a trivial procedure for MATLAB, given an appropriate algorithm.

The surface area of the region of interest is thus computed using the MATLAB function *surface_area*, which employs the cross product in order to determine the area of a triangle in three-dimensional space (cf. Appendices A and B for an explanation of the algorithm and its implementation using MATLAB code, respectively). This process can

be iterated to determine the area of each constituent triangle; by summing these individual contributions, the surface area of the entire region of interest can be computed.

By virtue of the nearest-neighbor index, which is determined using the Delaunay triangulation algorithm (as previously mentioned), the individual areas of the constituent triangles can be computed serially, but independently (i.e. it does not matter how the region of interest is traced, as long as the collected points are representative of the region of interest itself). This is because the Delaunay triangulation index (obtained using the *delaunay* function of MATLAB) defines the various triangles intuitively and independently (i.e. each index row defines one independent, constituent triangle), in a manner which is representative of the surface (provided that it the surface is traced appropriately). If, for argument's sake, the triangles were defined instead using the order in which the points are collected, and if the triangles' vertices were then defined using points (1, 2, 3), (2, 3, 4), (3, 4, 5), … , (*n*-2, *n*-1, *n*) (for *n* number of collected points), then the triangulation scheme would depend on the method of tracing, and thus would likely include overlapping triangles and other such problems (depending on how the region of interest is traced), and would therefore not be representative of the region of interest, thereby introducing error in the surface area calculation. Such a problem is thus resolved by employing the Delaunay triangulation algorithm in order to intuitively define these constituent, surface-approximating triangles using the nearest-neighbor points in the *yz*-plane (followed by extruding depth using the *x*-coordinate), and then determining their respective areas in a manner which does not depend on how the surface is traced (as long as the data points that are used are representative of the region of interest itself).

Appendix D summarizes the implementation of the aforementioned process using MATLAB code.

Design Alternatives

Our three designs all use the same data analysis using the previously mentioned Matlab procedure to find the surface area and to generate a 3D surface plot. Therefore, our designs focused on obtaining coordinates in the Cartesian system of the area of abnormality. In all the designs the patient must remain stationary or data will be skewed because points will be shifted in relation to each other. For large areas, such as those that cover the chest around the side and to the back, multiple areas must be calculated and added together.

A. Active Infrared Motion Capture

The active infrared motion capture system involves infrared motion capture cameras (PhaseSpace 1) and an LED that emits light in the infrared spectrum at 850 nm, as seen in Figure 3. Two camera units would be needed, each unit containing two cameras. The four cameras allow exact triangulation of the LED in three-space. The LED would be made into a stylus with the LED at one end. The software included in the system (Master) tracks the LED by measuring the distance to each camera. This software then outputs the data into a file that can be converted to a text document with the Cartesian coordinates.

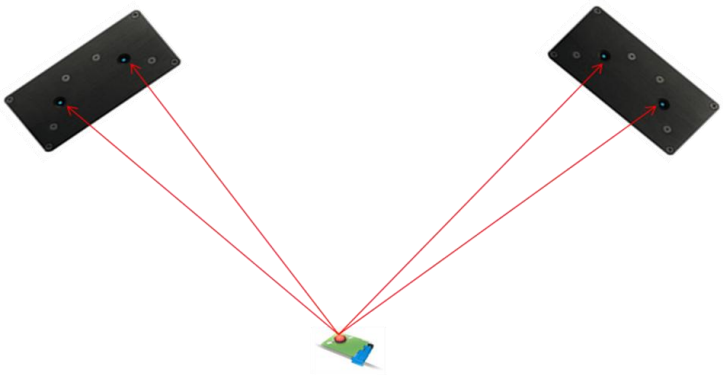


Figure 3: Active infrared system diagram

The active infrared system has two main advantages. The first of which is that it is very accurate. The cameras can measure distances with an accuracy of less than one millimeter. Also, with the active infrared system the collection of data points is quite easy. The system needs to first be calibrated after setup, but this process takes less than five minutes. After the system is calibrated, the clinician simply needs to outline the afflicted area with the stylus. This involves circling around the perimeter and moving the LED back and forth over the inside of the area to provide the points necessary to show the curvature of the surface (Figure 4).

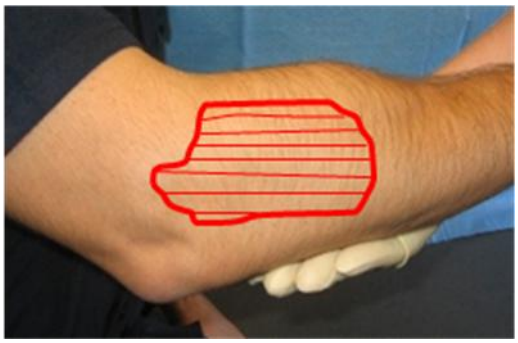


Figure 4: Example path needed to effectively map the area of sensory abnormality.

The active system carries some heavy. First and foremost, the cost of the system is staggering. A two camera setup including all necessary cabling and software would cost $17,560 after a 20% academic discount. The two other problems come from the

LED. First, it must be pointed at the cameras at all times. If it were to be tilted away or obstructed from the view of the camera by clinician or patient, the system would lose track of it and data points would be lost. This would lead to an incomplete map and inaccurate surface area calculations. Second, the LED must be moved across the patient's skin in order to map the curvature of the area. This can be problematic for patients that are hypersensitive where the slightest touch can cause extreme pain.

B. Passive Infrared Motion Capture

The passive infrared motion capture system involves infrared motion capture cameras and a small reflective ball (Figure 5). Three cameras would be needed to determine the position of the ball in three-space. Each camera has a ring of infrared LED's that emit light. This light is bounced off the reflective ball and is captured by the camera at the center of the LED ring. The reflective marker would be mounted on the tip of a stylus similar to the active system to allow the clinician to easily map the area. The included (OptiTrack 1) and add-on software, Point Cloud Toolkit, allows exportation of the position of the marker into a file that can be converted to a text document with the Cartesian coordinates.

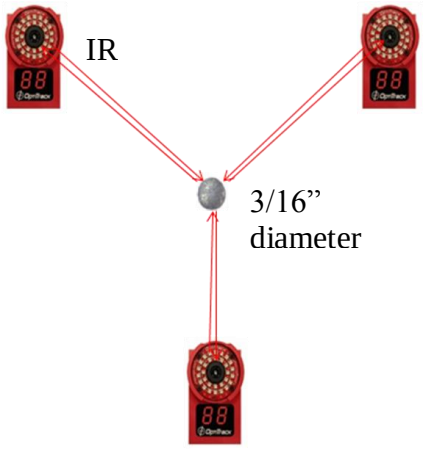


Figure 5: Passive infrared system diagram.

The passive system has the same advantages as the active system. It is highly accurate, capable of measuring distances with an accuracy of less than one millimeter. It is also easy to setup needing only a calibration to be ready to gather data. Once the system is set up, the afflicted area simply needs to be mapped in the same manner as the active infrared system. The key advantage by using this passive technology is the cost. For a three camera system, the minimum to map 3D points, the estimated cost is around $2,000. This includes the add-on software and necessary cabling.

The passive system also shares some of the same disadvantages as the active infrared system. While the cost is significantly less than that of the active system, it is still outside of our proposed budget. Also, the reflective marker still must be moved across the surface of the patient's skin, which is a concern for patient comfort.

C. Laser Distance Meter

The final design involves using a laser distance meter. To gather the points, an arbitrarily chosen point on the region of interest would be selected as the "origin". The distance would be measured to this point and saved for later. To determine the position of another point, the angles θ and Φ in the x and y directions respectively would be measured to the point as well as the distance to the point (Figure 6). Using the angles and the distances between the origin and the point being calculated, the Cartesian coordinates can be determined. By taking the origin distance times the sine of θ, the x coordinate relative to the "origin" can be found. The y coordinate can be found similarly by multiplying the point distance by the sine of Φ. The z coordinate can be calculated by using the cosines of the angles. The origin distance times the cosine of θ is subtracted

from the point distance times the cosine of Φ. This process would be repeated for each pointed needed to complete the surface map.

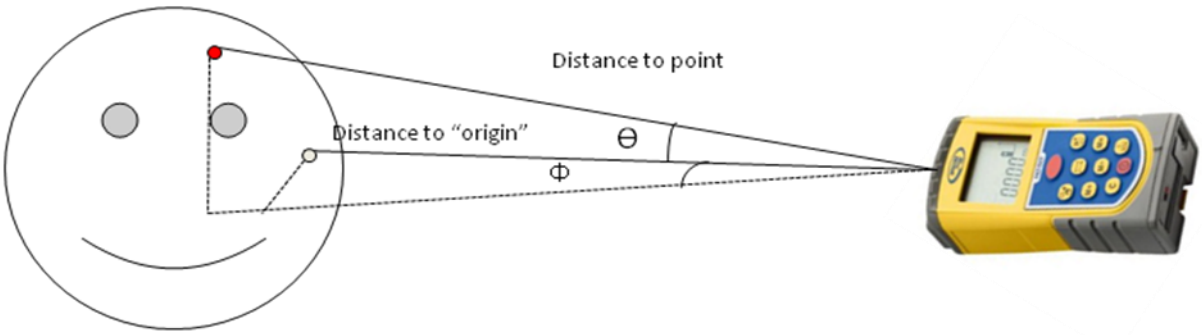


Figure 6: Laser distance meter

The laser distance meter has two advantages. The most significant is the cost. At only $130 (Spectra 1), it is well within our budget. The laser also risks no discomfort to the patient because it does not involve any patient contact.

This method has very big drawbacks, however. The biggest problem is the time and effort needed to gather the point data. Measuring the angles and the distances will take significant time. After being measured, there is no method to input it directly into a computer and would have to be entered by hand. The angle and distance data would then need to be converted into the points needed by Matlab to compute the surface area. Since the patient must remain still while the data is collected, the time needed to collect the data is unsatisfactory. The laser distance meter is also much less accurate than the infrared systems. It can only measure distances with an accuracy of plus or minus two millimeters. This is a problem when calculating the area of small regions of interest such as found on the face.

Design Matrix

	Possible Points	Passive IR	Active IR	Laser Meter
Accuracy	15	15	15	11
Ease of Use	10	8	7	2
Time	10	8	8	2
Cost: Initial	15	7	3	13
Repeatability	5	5	5	3
Patient Comfort	10	7	7	10
Total	**65**	**50**	**45**	**40**

Table 1: Design Matrix

The three design proposals were evaluated using a design matrix, seen in Table 1. The two factors given the most weight were accuracy and initial cost, which were the two main specifications that Dr. Backonja stated. The laser meter method scored the highest in cost, as the expenses would only total around $130. The infrared systems are considerably more expensive, and therefore did not fare as well in the category. They are, however, much more accurate than the laser meter method. Measurements using the infrared systems produce results to within a millimeter, which falls well within the specified 5 – 10 % accuracy with respect to the current tracing paper method. Using the laser meter, although still fairly accurate, introduces the factor of human error into the process. The clinician must make certain measurements and calculations by hand, which also makes the method slightly less repeatable because the clinician would likely measure a little differently each time. If just one error is made in the entire process, it will throw off the rest of the measurements and calculations as well.

The two main categories that eliminated the laser meter method were ease of use and time. It would take hours and hours of time for the clinician to carefully make all the correct measurements and calculations, and the process would likely be very difficult. It is for this reason that the method is impractical for this application. The process will probably extend beyond Dr. Backonja, so it is imperative that the chosen method be relatively easy to accomplish. Also, the patient must not move, so anything longer than a few minutes would cause the patient discomfort. The infrared systems relay the information instantly, and one receives feedback extremely quickly. No extra work is required by the clinician.

Between the two infrared systems, the main differences were in ease of use and cost. The LED on the tip of the pen in the active system must always be directly facing the camera, and this makes the passive system slightly easier to use, although the reflecting ball still must be in view of the camera. The cost of the two systems, however, was the key point in determining the desired design. Research has proven the active system to be upwards of $20,000, while the passive system is obtainable for $2000. The active system is well outside of our budget, while we may be able to afford the passive system. For these reasons, the passive infrared system is our chosen design.

Final Design

A. Hardware

The final design consisted of three OptiTrack FLEX:V100 cameras. These cameras will work either actively using infrared LED's or passively using reflective markers. As will be explained later, we used both of these features in our system. In order to use the cameras to track points in three dimensions, we also need to purchase special

software from OptiTrack called Point Cloud. The Point Cloud software is meant to be used as just to calibrate the cameras to gather data. Then the user can right their own programs to initiate, collect, and store data points as they desire using the included Point Cloud API (Application Programming Interface). The API is a set of built in functions that can be used with programming languages like C++ or C# to interface with programs. We did not use the API because we did not have time to write our own program and could gather raw data points using a feature in Point Cloud. The system needs a ground plane in order to finish the calibration process. This ground plane consists of three reflective markers arranged in a right triangle and is used to establish where the ground is in reference to the cameras. Instead of purchasing one from the manufacturer, we made our own using plywood scrap and markers we made using Styrofoam balls and reflective tape (Figure 7). We converted an LED pen into one that has an infrared LED (wavelength 850 nm) in the tip to gather the data points (Figure 7). This tool allows the user to easily move the marker around the region of interest.

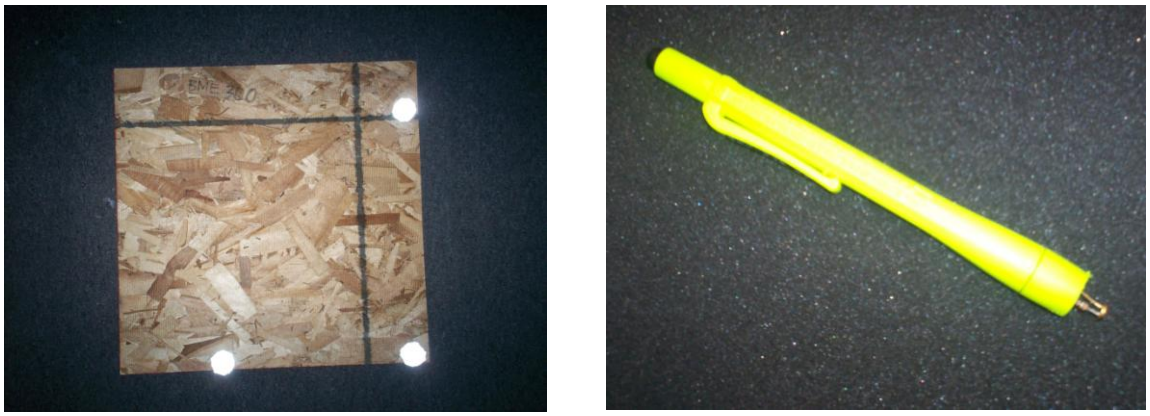


Figure 7: The ground square we constructed with homemade reflective markers (left), and the converted LED pen (right).

B. Setup

The cameras needed to be aligned in a large arc with the outermost cameras turned inwards for the best results. This arrangement was found to produce the best calibration results and most accurate data points. The three cameras are then connected to a PC using one USB cable for each camera. The USB cables from the cameras can also be plugged into a USB hub to allow many cameras to be attached to one computer. In order to synchronize the pulses of infrared light from each camera that is used with the passive features, a cable connects each camera to its neighbor. Therefore, one less cable is needed than there are cameras. The setup is shown below in Figure 8.

Figure 8: The three cameras set up for testing.

C. Calibration

The calibration process is completed using the Point Cloud software. First, the LED marker pen is moved around in the field of view of all three cameras while the system collects data. We found that for best results, the marker needs to be moved in each of the X, Y, and Z directions. After enough data is collected, the software calculates the positions of the cameras relative to each other. The ground plane is then placed on the ground in front of the cameras and set by simply clicking a button. Now that the calibration is completed, features of the software allow the user to display the volume that all three cameras can see and therefore collect accurate data. At this stage, the accuracy of the calibration can also be displayed by using two markers of known distance from each other.

D. Data Collection

The region of interest must first be marked out on the patient before the data collection can begin. The data points are gathered all at once by moving the marker around the perimeter of the area and then back and forth across the interior of the surface to map the curvature of the surface. If the interior surface is not mapped, the calculation algorithm will connect points across the surface. These triangles will cut through the body part and give an inaccurate area. The software is told to start recording points and the marker is moved in the fashion described. After the entire area is covered, the user simple hits stop. The data can be saved from the software and is stored in an Excel spreadsheet. This spreadsheet is then loaded directly into the Matlab script to calculate the surface area and visually display the region of interest with the collected points and triangles used in the calculation.

Future Work

In order for the system to be more conducive to clinical use, a number of improvements could be added in the future. First, it is extremely important that the cameras do not move once the system is calibrated. If they are moved even the slightest distance, everything will have to be recalibrated. For this reason, a method of mounting the cameras must be developed. This could be anything from a set of three tripods to simply mounting the cameras on the wall. The result would be less work for the clinician and a more accurate system. Also, the OptiTrack software comes with several built-in functions that could greatly increase the effectiveness of our system. For example, it allows the user to turn down the frame rate from 100 frames per second to 75, 50, or 25 frames per second. This could potentially give more accurate results since the points would not be so clustered. In order to access the built-in functions, however, one must tap into the code of the software, using such programming capabilities as C++ or C#. Once the programming is solved, however, many settings would be able to be changed on the cameras, allowing the system to be set up in a way that best suits our situation.

For future testing, the main challenge we would face is determining the best method of tracing the affected area on the skin. For example, should the clinician trace the border first, and then go back and forth across the area? Through testing of various methods, we would find the technique that gives the most accurate surface area. We also would attempt to create an algorithm that would allow us to calculate the exact surface area. Anything that can improve the accuracy of our system is crucial. Next, we would most likely need to refabricate the LED stylus. As it is right now, it is too easy for the LED to move out of the line of sight of the cameras. When this happens, the software

freezes on the last point that the cameras recognized. Once they pick up the LED again, the marker on the screen jumps to the new location, causing obvious problems with the data set. The LED on the new stylus needs to be more visible so that the cameras will recognize it, regardless of the positioning of the stylus. Finally, to make the whole system compatible in a clinical setting, we would have to perfect the GUI that we began to create this semester.

Ultimately, the remaining work on this project mostly includes making the system easy for a clinician to use. Dr. Backonja wanted something easier and more accurate than his tracing paper; it would not make sense for him to receive a system that is relatively complicated to operate and could potentially give poor results. This is its current state. However, with another semester's worth of improvements, our infrared motion tracking system could easily become an incredibly valuable tool that could save many clinicians a great deal of time and effort.

Potential Problems

One of the main problems that could result from the passive infrared system involves our budget. We were initially given a budget of $500 - $1000, yet three infrared cameras would cost around $2000. If our budget is unable to be expanded, we will not be able to purchase the cameras and will resort to simply using a lab on campus. However, it appears at this point that our budget will be able to support the purchase of three cameras. Another problem is the fact that the patient is unable to move his/her body during the whole procedure. Because the process is relatively quick, this should not cause any discomfort to the patient. However, it does make it impossible to map two separate sides of the body because the pen must always be in the sight of the cameras.

This could be solved by mapping one side of the body, saving the information, then mapping the other side of the body. The total surface area could then be easily calculated. Another option, if our budget was expanded even further, would be to purchase a fourth camera. When positioned correctly, this could potentially allow the clinician to map both sides of the body at once.

Testing

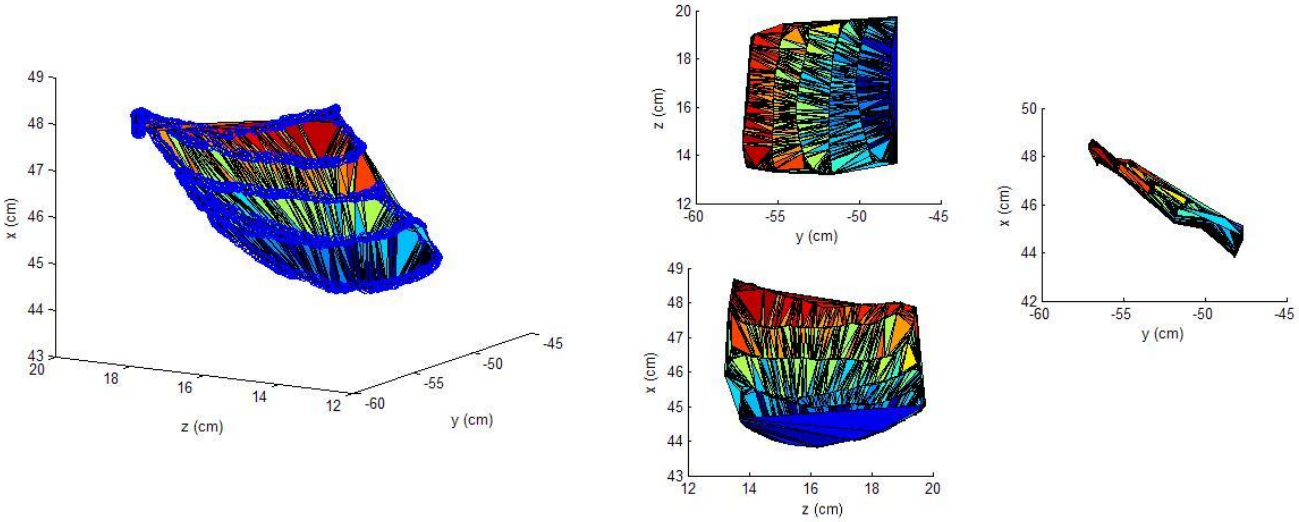
Once the cameras were received, they had to be correctly calibrated before any measurements could be taken. The cameras were placed on the top of a table in a straight line with the end two cameras tilted slightly in to allow the fields of view to intersect. Each time the cameras were calibrated in this manner the orientation of the markers of the calibration square was distorted. Having the cameras too close together inhibited to computer's ability to triangulate the system and caused it to return invalid results. In order to obtain a good calibration, it was necessary to spread the cameras out further and have the two end cameras turned in at a more dramatic angle. In order to solve this problem each camera was placed on a separate surface approximately a meter apart.

After the cameras were properly calibrated and the Matlab script was written, testing was needed in order to determine the best way to collect the points and how accurate these measurements were. The first tests which were conducted traced both a two dimensional rectangular surface and a portion of a cylindrical can in various ways. Different ways to trace the surface included outlining only the border, weaving across the interior of the surface, and finally doing both the border and the interior in the same trial. The surface area calculations of these initial trials were very inaccurate for a number of reasons. The first problem was that the algorithm which was used in the script would

create the triangles of the surface in sequential order rather than according to natural neighbors. This caused gaping holes in the surface where the points had not aligned correctly and did not create a representative surface (Appendix A). By changing the algorithm which was used, it created a much more representative surface and therefore gave a much better surface area calculation.

Another problem with the initial trials was the orientation of the surface with respect to the cameras. During calibration the coordinate system of the environment is set and in this case the floor was set at the XY plane with the positive Z axis coming up out of the floor. When the cylinder was traced it was placed upright on the floor, in the YZ plane. However, when the surface was being interpolated it was still being done so in the XY plane. This inconsistency created a distorted surface and made for inaccurate surface area calculations. With the XY plane changed to the YZ plane in the script it was necessary to have the surface being traced perpendicular with the cameras, which produced much more accurate results.

Once these problems had been dealt with, the original trials could be re-interpolated. The new surfaces had areas which were within the 5-10% accuracy range given by the client. As Figure 9 shows, the actual surface area of the curved surface was 70.0 cm^2 and it was calculated as 73.4 cm^2. This surface area is within 4.9% of the actual value. Another curved surface was made by outlining a known area of 22.9 cm^2 on a team member's arm. The area which the program calculated was 25.44 cm^2 which was within 10.4% of the actual value.



A rectangle (placed on the forearm). Three planar perspectives and a general three-dimensional perspective (left-most), which includes the collected data points (blue), are shown. The actual area of the rectangle was determined to be 70.0 cm^2, while the computational algorithm determined the area of the rectangle to be 73.4 cm^2.

Figure 9: Interpolated Surface

Works Cited

BurnCase 3D: 3D Burn Documentation for Professionals. 18 Feb 2003. RISC Software. 15 Oct 2008. [http://www.burncase.at/index.php?page=home].

Delaunay Triangulation. 2008. The MathWorks. 12 Oct 2008. [http://www.mathworks.com/access/helpdesk/help/techdoc/ref/delaunay.html].

Neuropathic Pain. Feb 2007. Merck &Co. 14 Oct 2008. [http://www.merck.com/mmpe/sec16/ch209/ch209c.html].

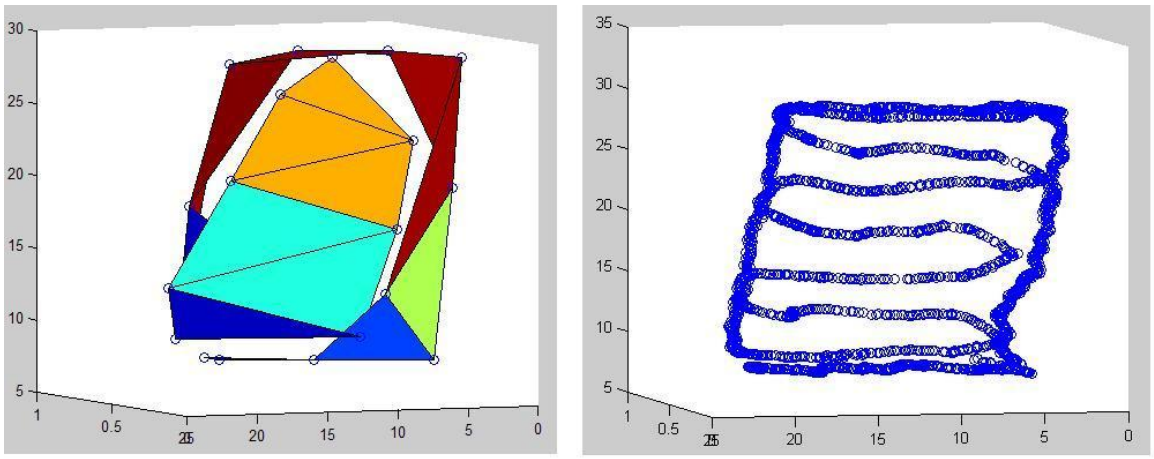
OptiTrack FLEX. 23 June 2005. OptiTrack. 13 Oct 2008. [http://www.naturalpoint.com/optitrack/products/flex-v100/].

Pain Management: Neuropathic Pain. 13 Dec 2007. Medicine Net. 14 Oct 2008. [http://www.medicinenet.com/neuropathic_pain/article.htm].

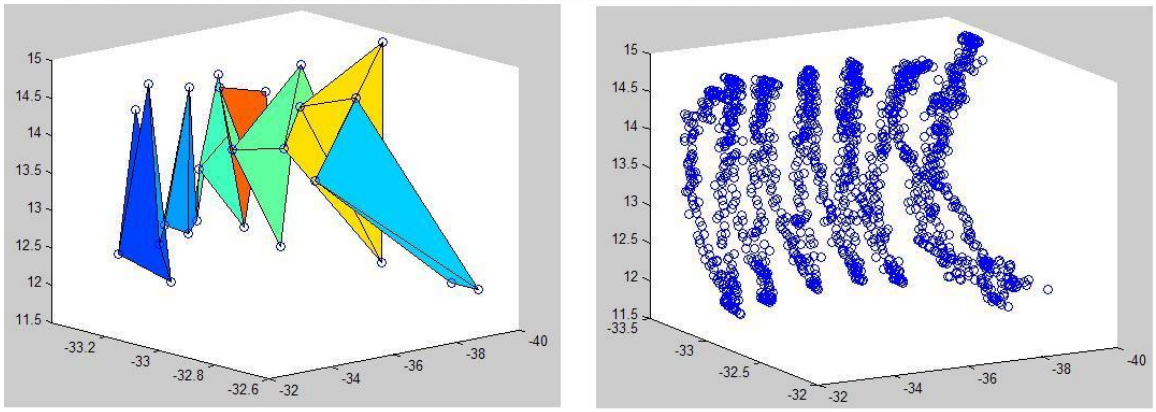
PhaseSpace Motion Capture. 2006. PhaseSpace. 9 Oct 2008. [http://www.phasespace.com/productsMain.html].

Spectra Precision HD50. 20 March 2007. Trimble. 6 Oct 2008. [http://www.trimble.com/hd50.shtml].

Appendix A: Original Matlab Algorithm Test



A rectangular base. The tracing scheme is shown on the right, while the computational scheme is shown on the left (using every 125th point of the tracing scheme). The actual area of the rectangular base was determined to be 385.86 cm^2, while the computational scheme (on the left) calculated the area of the surface to be 373.82 cm^2.



The side of a cylindrical can. The tracing scheme is shown on the right, while the computational scheme is shown on the left (using every 70th point of the tracing scheme). The actual area of the region (a rectangular portion of the side of a cylindrical can) was determined to be 27.82 cm^2, while the computational scheme (on the left) calculated the are of the surface to be 10 cm^2.

Appendix B: The Cross Product Algorithm

Consider three arbitrary points located in various positions in three-dimensional space. After assigning each of the three points the respective names $P_i(x_i, y_i, z_i)$, $P_{i+1}(x_{i+1}, y_{i+1}, z_{i+1})$, and $P_{i+2}(x_{i+2}, y_{i+2}, z_{i+2})$, let these three points form the vertices of a triangle (in space). Two three-dimensional vectors can now be defined as follows: $\overrightarrow{P_iP_{i+1}} = \left(x_{i+1} - x_i, y_{i+1} - y_i, z_{i+1} - z_i\right)$ and $\overrightarrow{P_iP_{i+2}} = \left(x_{i+2} - x_i, y_{i+2} - y_i, z_{i+2} - z_i\right)$, using $P_i(x_i, y_i, z_i)$ as the initial point for both vectors. The cross product of these two vectors is then determined as follows:

$$\begin{aligned}\overrightarrow{P_iP_{i+1}} \times \overrightarrow{P_iP_{i+2}} &= \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ x_{i+1} - x_i & y_{i+1} - y_i & z_{i+1} - z_i \\ x_{i+2} - x_i & y_{i+2} - y_i & z_{i+2} - z_i \end{vmatrix} \\ &= \left[\left(y_{i+1} - y_i\right)\left(z_{i+2} - z_i\right) - \left(y_{i+2} - y_i\right)\left(z_{i+1} - z_i\right)\right]\hat{i} \\ &\quad + \left[\left(z_{i+1} - z_i\right)\left(x_{i+2} - x_i\right) - \left(z_{i+2} - z_i\right)\left(x_{i+1} - x_i\right)\right]\hat{j} \\ &\quad + \left[\left(x_{i+1} - x_i\right)\left(y_{i+2} - y_i\right) - \left(x_{i+2} - x_i\right)\left(y_{i+1} - y_i\right)\right]\hat{k}\ .\end{aligned}$$

Using this cross product, the area of the triangle formed by the three points can then be determined as one-half of the magnitude of the cross product: $A_{\triangle} = \frac{1}{2}\left\|\overrightarrow{P_iP_{i+1}} \times \overrightarrow{P_iP_{i+2}}\right\|$.

Therefore, if given a matrix **B** containing *n* number of such vertices of triangles:

$$B = \begin{bmatrix} x_1 & y_1 & z_1 \\ \vdots & \vdots & \vdots \\ x_i & y_i & z_i \\ x_{i+1} & y_{i+1} & z_{i+1} \\ x_{i+2} & y_{i+2} & z_{i+2} \\ \vdots & \vdots & \vdots \\ x_n & y_n & z_n \end{bmatrix},$$

this process can be iterated to find the total area of the surface composed of the *n-2* triangles formed by these vertices.

Appendix C: A MATLAB Function which Implements the Cross Product Algorithm

```
function cross_product = surface_area(A)
% This function assumes that the input argument "A" is a matrix of row vectors representing
% points Pi(xi,yi,zi), i.e. A = [x1 y1 z1; x2 y2 z2; ... ; xn yn zn] for points i=1 to n;
% the resulting matrix (A) can be though of as three column vectors representing all x, y,
% and z components, i.e. A = [x y z], where x,y,z are column vectors.
%
% So, as a MATLAB Command Window output, the matrix/variable "A" (which is presumably
% obtained by importing an excel spreadsheet containing data points with three columns
% corresponding to x,y,z data for each point) should appear as follows:
%
%A =
%
%    x1    y1    z1
%    x2    y2    z2
%    .     .     .
%    .     .     .
%    .     .     .
%    xn    yn    zn
%
% which represents points P1(x1,y1,z1) to PN(xn,yn,zn).

cross_product = 0; % initialize the counter

for i = 1:length(A(:,1))-2 % iterates i=1 to i=n-2 points (see explanation below)

% i component of cross product:
term1 = ((A(i+1,2)-A(i,2))*(A(i+2,3)-A(i,3)) ...
    - (A(i+2,2)-A(i,2))*(A(i+1,3)-A(i,3)));

% j component of cross product (note: there is a "-" in the determinant for
% the j term, so the subterms are reversed accordingly, in order to follow
% a "+" convention):
term2 = ((A(i+2,1)-A(i,1))*(A(i+1,3)-A(i,3)) ...
    - (A(i+1,1)-A(i,1))*(A(i+2,3)-A(i,3)));

% k component of cross product:
term3 = ((A(i+1,1)-A(i,1))*(A(i+2,2)-A(i,2)) ...
    - (A(i+2,1)-A(i,1))*(A(i+1,2)-A(i,2)));

triangle_area = (1/2)*sqrt(term1^2 + term2^2 + term3^2);
% the area of a triangle is one-half of the magnitude of the cross product
% (which represents the area of the parallelogram formed by the two
% vectors whose cross product is taken; in this case, the two vectors are:
%
% v1 = (x(i+1)-x(i),y(i+1)-y(i),z(i+1)-z(i))
%
% v2 = (x(i+2)-x(i),y(i+2)-y(i),z(i+2)-z(i))
%
% which is computed for points i=1 to i=n-2, since the last triangle will
% be made using the third-to-last point "(x(i=n-2),y(i=n-2),z(i=n-2))" as
% the last vector pair's (v1(i=n-1),v2(i=n)) initial point.)

    cross_product = cross_product + triangle_area; % update the counter
end
```

N.B. This function is called in the data analysis script (See Appendix D, line 36 of the MATLAB script).

Appendix D: A MATLAB Script which Performs the Data Analysis

```
% This script has been prepared by the Sensory Mapping team (BME 300/200, Fall 2008)
% consisting of: Jeremy Schaefer, Colleen Farrell, Steve Wyche and Adam Pala (University of
% Wisconsin-Madison, Department of Biomedical Engineering). Special Thanks to Dr. Julie C.
% Mitchell of the Deparment of Mathematics (University of Wisconsin-Madison) for helping with
% the code (namely determining the vertices of each nearest-neighbor triangle in in order to
% calculated surface area using each triangle independently [cf. lines 33-35 of this script])

clf % clears plot ("Figure") from previous usage
clc % clears screen ("Command Window") from previous usage
clear % clears variables ("Workspace") from previous usage

initial = input('Type in the data file name (including .csv): ','s');
ROI = input('Describe the region of interest: ','s');
A = load(initial); % loads the file (user-specified) containing data points
                   % collected using the OptiTrack cameras/Point Cloud software

% Columns 3,4,5 of matrix A correspond to coordinates x,y,z (respectively)
% of each data point, according to how the hardware captures and stores the
% point(s) (in a .csv file); furthermore, the scaling factor "100" is used
% to convert all spatial dimensions to centimeters (for matrix B, shown below).
B = [A(1:length(A),3) A(1:length(A),4) A(1:length(A),5)]*100;

x = B(:,1); % "depth" dimension (the "normal" to the surface)
y = B(:,2); % planar dimension (used for initial 2D Delaunay triangulation)
z = B(:,3); % planar dimension (used for initial 2D Delaunay triangulation)

TRI = delaunay(y,z); % finds the natural neighbors in the yz plane (x is "normal" to surface)

SA = 0; % initialize the counter (this keeps track of the surface area, which is calculated
        % for each triangle during the iterations, and is thus summed to find the total
        % surface area of the region of interest based on the triangulation scheme)

for i = 1:length(TRI(:,1))
    ind = TRI(i,:);
    myvec = [x(ind(1:3)) y(ind(1:3)) z(ind(1:3))]; % contains the vertices of each triangle
    surf_area = surface_area(myvec); % calculates the surface area of each triangle
    SA = SA + surf_area; % updates the counter for each iteration
end

% ------------------------------------------------------------------------
% Dislay the surface area calculation as follows:
disp(' ') % leave a space to prevent crowding of text
disp(['The calculated surface area of the ' num2str(ROI) ' is ', ...
    num2str(SA),' squared centimeters'])
disp(' ') % leave a space to prevent crowding of text

% ------------------------------------------------------------------------
% Plot the points and triangulated surface:
hold on
plot3(y,z,x,'o'), trisurf(TRI,y,z,x) % plots the points and surface
title(num2str(ROI)) % assigns the "region of interest" as the plot title
xlabel('y (cm)'),ylabel('z (cm)'),zlabel('x (cm)') % labels the axes
hold off

% ------------------------------------------------- %
% Script last updated on: Thursday December 04, 2008   %
% ------------------------------------------------- %
```

N.B. Line 14 of this MATLAB script (annotated in the margin) imports the .csv file from the data collection trial (this is the output from the Point Cloud software), whose *x*-, *y*-, and *z*-coordinates are extracted as column vectors and scaled to centimeters in Line 21. Also note that Line 36 calls the *surface_area* function (See Appendix C).

Appendix E: Product Design Specifications (PDS)

Sensory Mapping

Client: Miroslav Backonja

Advisor: Mitch Tyler

Team Members

Colleen Farrell (BSAC)

Adam Pala (BWIG)

Jeremy Schaefer (Team Leader)

Steve Wyche (Communicator)

Function:

The purpose of this project is to design a systematic method to accurately measure the surface area of sensory abnormalities.

Client Requirements:

1) **Should give quantitative results**
2) **Should be able to measure contoured areas of the body**
3) **Results should be reproducible**

Design Requirements:

1. **Physical and Operational Characteristics**
    a. *Performance Requirements*: Measure more accurately than the current tracing method. Must function over all surfaces of the body
    b. *Safety*: Must have minimal skin contact due to patient sensitivities.
    c. *Accuracy and Reliability*: Accurate within 5-10% of actual surface area. Results should be reproducible. The cameras to be used can measure distances with an accuracy of less than one millimeter
    d. *Life in Service*: Device should last through the duration of the clinical trials
    e. *Shelf Life*: Indefinite, since the final product will not likely include disposable parts or perishable materials. Batteries in LED penlight will need to be replaced when they die
    f. *Operating Environment*: Function at room temperature

g. *Ergonomics*: Must be comfortable and convenient for doctor use
h. *Size*: No strict guidelines. Small enough for in clinical use and to allow easy portability
i. *Weight*: The pen with infrared LED will weigh less than a pound
j. *Materials*: Infrared motion capture equipment. Including: three infrared cameras, infrared LED pen, ground plane, USB cables, synchronization cables
k. *Aesthetics*: Due to use in clinical testing, aesthetics are not of utmost concern

2. Production Characteristics

a. *Quantity*: one working prototype
b. *Target Product Cost*: $500-$1000 for working prototype

3. Miscellaneous

a. *Standards and Specifications*: The prototype will be compared to the current tracing method that is being employed
b. *Patient-Related Concerns*: Minimal skin contact
c. *Competition*: Zcorp Zscanner 800