DEVELOPMENT AND ANALYSIS OF DERMAL WOUND IMAGE PROCESSING TECHNIQUES USING CHAN-VESE AND EDGE ACTIVE CONTOUR METHODS

Peter J. Mueller

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Master of Science in the Department of Biomedical Engineering.

Chapel Hill
2017

Approved by:

Devin Hubbard

Dwight Walker

Shawn Gomez

Gianmarco Pinton

ABSTRACT

Peter J. Mueller: Development and Analysis of Dermal Wound Image Processing Techniques Using Chan-Vese and Edge Active Contour Methods
(Under the direction of Devin Hubbard)

Present methods for evaluation of burn wounds rely heavily on qualitative and Total Body Surface Area (TBSA) estimations. Herein, a digital method for calculating the surface area of burn wounds is proposed as a useful tool for monitoring and measuring changes in burns. This study tested two segmentation methods: a statistical analysis technique, and an active contours technique using edges and Chan-Vese. All methods were tested on images of burns taken from a DSLR camera, and Microsoft Kinect V2 and compared to digitally drawn traces of the wounds. Using Dice's Coefficient as a measure for agreement between masks, the DSLR images resulted in agreeable segmentations (D=0.939 for edge, D=0.9362 for Chan-Vese), while images taken with the Kinect did not meet the threshold for agreeability (D=0.815 for edge, D=0.819 for Chan-Vese). This testing shows the active contours method is the plausible method for characterizing high-resolution color burn wounds.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

CSV	Comma Separated Values
DSLR	Digital Single-Lens Reflex Camera
IR	Infrared
LED	Light Emitting Diode
NIR	Near Infrared
RGB	Red, Green, Blue
TBSA	Total Body Surface Area

Chapter 1

Introduction/Background

Introduction to Burns

Burn wounds are traditionally classified into 3 degrees according to the burn depth. 1st degree, or superficial burns only affect the epidermal (outer most) layer of the skin. The appearance of a superficial burn consists of a red, moist lesion which tends to resolve spontaneously with little to no medical intervention. Partial thickness, or 2nd degree, burns are those that include both superficial partial thickness burns as well as deep partial thickness burns. Superficial partial thickness burns affect both the epidermis and the upper dermis. Superficial partial thickness burns are typically capable of healing on their own, although infections or drying out of the wound can cause progression to a deep partial thickness wound. Deep partial thickness wounds affect both the epidermis and extend deep into the dermis. These deeper wounds may not heal spontaneously and may require a warm, moist environment in order to recover. Full-thickness, 3rd degree burns are those in which all layers of the dermis are affected. Full-thickness burns are incapable of healing on their own, requiring excision and grafting of new skin.

It is important to note that many burns consist of a combination of these different degrees of depths. Difficulty lies in assessing deep partial thickness burns, as they may initially appear and be diagnosed as superficial partial-thickness. Such complicated wounds require a follow-up examination 48 hours after the initial examination to determine the exact depth of the wound [1], [2]. Currently in digital analysis of these wounds, only surface features of the wounds are considered, including the analysis discussed in this paper. Opportunity exists in studying the subdermal features of the wounds, however deep features are generally more difficult to image.

The features of the local surface response of burns consist of three different zones that make up the area of an individual burn wound: coagulation, stasis, and hyperemia. The zone of coagulation is the area of the wound where the damage is maximal, and is generally the central region of the burn where tissue loss is irreversible. Surrounding is the zone of stasis, a region in which tissue is capable of being recovered. It is a goal of wound recovery to prevent the zone of stasis from converting to the zone of coagulation. The most peripheral area of the wound is the zone of hyperemia, the region that is unburned but appears red due to increased blood flow in the area [3].

The current method of burn wound size assessment involves calculating the TBSA of a patient [4]. This is done by using predetermined charts that list area estimations for different parts of the body, such as the Rule of Nine's or Lund and Browder charts. A study done by T.L. Wachtel [5] compared these methods and found significant amounts of variation between the two methods, as well as variations within use of an individual method. The methods also only produce an area estimate in percentage of body surface area, which, while useful clinically, does not provide adequate information regarding absolute surface

area. Absolute values are necessary for accurate measurements and direct comparisons. For example, while attempting to analyze data for a clinical trial comparing two wounds that may only have a centimeter of difference between them, the currently used broad estimator is not accurate enough to calculate this difference.

The rise of technological advancements in computer systems has created a new avenue for wound size assessment namely: computer assisted calculation [6]. These computational processes usually involve digital photography of the wound in either 2-dimensions (2D) or 3-dimensions (3D), and then application of an algorithm to analyze the image and calculate the surface area. Traditional photography techniques include digital single lens reflex (DSLR) photography, however other techniques are emerging. For example, the Microsoft Kinect 2.0 camera is capable of taking color, near infrared, and depth images.

Herein we propose the development and testing of an analysis program for automating wound measurements. We hypothesize that a statistical analysis and active contours segmentation of the wound will be able to accurately quantify wound size to an absolute value. The Microsoft Kinect 2.0 Camera has been proposed as a possible tool of image acquisition for the computer assisted calculation of burn wound size. We hypothesize that the Kinect is capable of acquiring images that can be analyzed for accurate burn wound measurements.

Digital Photography in Dermatology

Outside of direct patient contact, digital photography plays an important role in dermatology as it serves as the primary means of analyzing the appearance of a wound. Obtaining a color, shape and intensity-accurate image of a wound is important to rendering a correct diagnosis. The DSLR camera is the current recommended camera for taking dermatological images, although traditional compact cameras are still used in some cases due to ease of use and cost. The DSLR is preferred due to the camera's high pixel resolution, modular lenses, and the suite of customizable features present in the device [7]. Lightning and flash play an important role in the acquiring of the images. These cameras, as well as film, are only capable of resolving approximately 6 orders of magnitude of light intensity. This figure is significantly lower than the human eye's 26 magnitude capability, this makes cameras more prone to errors from lighting. Taking an image of a subject that is too brightly illuminated, whether from a flash or external light source can cause data loss or make the subject appear whiter than it appears in ideal lighting. The same can be said for taking an image under insufficiently light intensity. Uneven lighting can also be detrimental as a solid color object could appear as a gradient. Insuring ideal lightening conditions are met are absolutely critical for accurate diagnosis of wounds, especially in images of wounds taken at different times.

The Microsoft Kinect 2.0 Camera is a camera that was packaged with the Microsoft Xbox One gaming system released in November 2013. This camera is capable of taking video in 3 different formats: RGB, near infrared (NIR), and 3D time of flight (TOF). The Kinect captures video of all three channels at the same time, recording all three at 30 frames per second. The RGB camera has a pixel resolution of 1980 x 1080 and functions as a

traditional digital camera. The NIR camera has a pixel resolution of 512 x 424 and has an optical pass band wavelength of 850-860 nanometers [8]. The NIR camera also makes use of infrared LEDS (˜860 nm) in the device to illuminate the subject with infrared light. The infrared channel is of interest in the dermatological field due to its ability it image subcutaneous vasculature as shown by Vladimir Zharov [9]. Due to blood's ability to absorb NIR light in higher quantities than surrounding material, such as fat or skin, superficial vasculature is easy to visualize due to the high contrast with surrounding material. The depth channel also makes use of the infrared camera using ToF technology to create a depth profile of the image. The ToF sensing works by comparing the phase of the initial IR wave that is sent out via the IR LED's to the phase of the received IR wave that is bounced from the object to the depth sensor. This process is done using multiple frequencies to insure accuracy [10] [11]. The depth images that are created are 11-bits in depth which corresponds to approximately 1mm spatial resolution over the full operational range (˜8 m) [8]. Due to this unique combination of color channels, the Kinect was chosen to be studied as a possible new tool for acquiring dermal wound images.

When capturing an image with a digital camera, an analog to digital conversion takes place. Traditionally, the RGB color space is used to digitally represent the analog colors. Each pixel has a Red, Green, and Blue component with 0,0,0 representing black, and 1,1,1 representing white, using a scale of 0-1 (another common scale is 0-255). This color space is useful for displaying colors, but not as useful for interpreting colors [12]. The human eye does not perfectly separate the color spectrum into R, G, and B channels like the RGB model. New models have been created to better emulate how humans interpret color. One of these models is the L*a*b color space. In this space, L is lightness, while a and b are color opposite

dimensions, yellow-blue, and red-green. This space is ideal for image analysis, because reducing the gain of the L channel can balance the effect that lighting in the images causes on color values [12].

Changes in the illumination of a scene can result in changes in perception of color. This is true for both imaging and human vision. When directly comparing image intensity values, it is important to ensure that colors are represented with the same value. White balancing is a method for attempting to correct this color offset. White Balancing involves selecting a white portion of the image. The RGB pixel intensity values at this known white point are then used to scale the rest of the pixels in the image with the known white values [13][14]. Many DSLR camera have an automatic white balance function, however it is possible to do a manual white balance by taking images of white cards in the same illumination as the subject.

Herein images taken of burn wounds from a DSLR and Xbox Kinect camera will be tested with developed methods of image segmentation. These methods will attempt to calculate an absolute surface area value for the wounds imaged. The images and results from the DSLR and Kinect cameras will be compared to assess the Kinect's ability as a dermatological image acquisition tool.

Chapter 2

Statistical Analysis Program

The first attempt to analyze the burn wound images is a Statistical Analysis program. This program is designed to acquire surface area estimates of the wound being analyzed by taking advantage of pixel color intensity. Its goal is to measure skin whose color is abnormal compared to the surrounding skin. The statistical filter is based on a previously designed red dominance filter that creates a mask of pixels in the image whose red values are dominant. A mask is a binary image whose pixel values are either one or zero. If a pixel meets the 'true' condition of the filter or algorithm, it is given the value one in the mask. Pixels whose red values are at least 40% of the maximum pixel intensity larger than the combined green and blue values are considered dominant and are given a value of one in the mask. Measuring redness does not capture all of the necessary color information in the image. Colors such as white, brown, and black are often present in burns, but are not designed to be picked up by this red dominance filter. The statistical analysis program is designed to measure all abnormal pixel intensities for all three color channels. Images from both DSLR and Kinect cameras were recorded to compare accuracy in results.

Data Collection

In early 2016, GlaxoSmithKline (GSK) performed a study taking images of burn wounds. GSK then provided us with the images from this study to develop and test different burn analysis methods. The subject of these images was a series of eight circular burns administered to groups of mini pigs. Two groups (A and B) of six pigs each were used in the study. Each pig had four burns on the left, and four burns on the right of their backs. Wounds were labeled L1-4 and R1-4, the letter determining what side of the pig it was on, and number counting down starting at the head. Images of the pigs were taken with three imaging modalities: a Canon EOS REBEL T5 DSLR camera fitted with a Canon EF-S 18-55 mm lens, a Silhouette Star camera, and a Microsoft Kinect 2.0 camera. Images were acquired both before and after the initial burn, and every two to three days after, up to 30 days for Kinect images and 40 days for DSLR and Silhouette images. Silhouette images were not used in this study, as the images generated have alternative lighting conditions as well as artifacts from internal processing.

The method of image acquisition for the DSLR camera consisted of a clinician holding the camera without assistance (tripod, stabilizer, etc.). The subject of the image was an individual wound from the series of eight on the pig, and a section of a ruler located above the wound for scale. Eight images were taken for each pig for each time period. For the Kinect, the camera was plugged into a laptop that was used to direct image acquisition. The Kinect images were extracted from short videos of the subjects (this is currently the only way to record data with the Kinect 2.0). The camera itself was held by a clinician at a distance of about one meter from the pig. Due to the large minimum focal distance and wide viewing angle of the Kinect, images are wider and zoomed out. The Kinect has a minimum imaging

distance of 0.5 meters for the NIR camera, hence images taken closer than 0.5 m appear very bright and contain little to no usable data. The Kinect images were taken of the entire left or right side of the pig, consisting of four wounds per image. Kinect images were taken at two different distances to accommodate the RGB and NIR cameras.

Pre-Processing

DSLR images were received in .JPG format with 5184 x 3456 pixel resolution. The .RAW image files were not saved by the study team. No pre-processing was required for these images. The Kinect images were sent as .XEF files, which are specific to Kinect videos, and contain four channels RGB, NIR, Depth, and Body. The body channel (unused in this experiment) is the location of 'body shapes' if the Kinect detects any in the depth frame. The only default program to read .XEF files is the Kinect Studio 2.0 program included in the Kinect APK. Kinect Studio 2.0 does not contain a feature for exporting images or stills of the videos. CK Imaging is a program developed in collaboration with GSK for the purpose of capturing and viewing Kinect Videos, as well as exporting the individual frames, a function not available in Kinect Studio 2.0.

To export frames from a Kinect .XEF file, the data must first be loaded into CK Imaging. This can be done by opening a file with the included file browser, or by acquiring the images directly through CK Imaging. If opening a previous recording, the Preprocess File option must be selected as well as the desired channel for output (RGB, Depth, IR, Body). In the settings menu the number of frames to be exported can be changed. The frames exported are from the first frame to the nth frame, where n is the chosen number of exported frames.

Originally the frames were exported in .CSV format, consisting of a spreadsheet with row and columns representing X and Y pixel location values, with three values in each cell representing the R, G, and B pixel values. In the case of NIR and Depth images, only one value is in the cell, representing intensity. The functionality to export images as .PNG image files was later added to the program with the option to export the .CSV files. RGB videos taken from the Kinect are exported as 1920 x 1080 pixel color .PNG images. NIR/Depth videos are exported as 512 x 424 pixel grayscale .PNG images. These depth images are the same resolution as the original video recordings [8]. The Kinect files from the study were exported using CK Imaging's .PNG export feature and analyzed in Matlab. The first clear, sharp, in-focus frame in each file was used as the image to be analyzed. For RGB Kinect images, the close-distance Kinect videos were used. For NIR/Depth images, the further distance Kinect videos were used. This was because the NIR/Depth channels in the original distance videos were commonly "blown out," consisting of areas of the image where all the pixels' intensities are maximum, containing little to no usable data. A screenshot of the CK imaging program interface is shown in Figure 1.

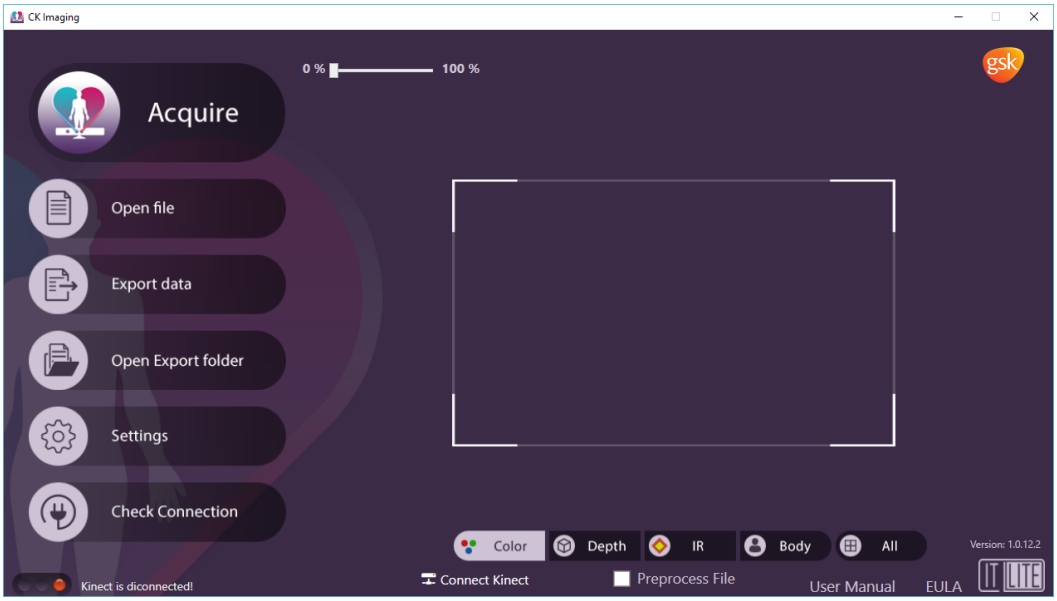


Figure 1. Screenshot of CK Imaging. Used to acquire and view Kinect videos and export them as single frames

Statistical Algorithm

Building on the previous work that used a redness threshold filter, Matlab (Mathworks, Inc. Natick, Ma) was used to design a new program that makes use of all of the RGB pixel intensities in the wound to try and interpret wound location. Instead of comparing the RGB values against each other, like in the red dominance filter, this program compares them against the RGB values of the skin. Unlike the red dominance filter, the statistical analysis program requires user input to operate. To start, the user determines the pixel density scale of the image by moving two ends of a line drawn over the image to line up to points of a known distance apart (ex: a ruler). Next, two rectangular patches of skin next to the wound are selected by the user to be used as the average skin values. Two areas were used in an attempt to correct for uneven lighting in some of the images. An ideal sample would be an area of the skin that consists of even lighting (no shadows or changes in intensity), such as being bright or dark relative to the rest of the image, as well as free of blemishes or other

abnormal conditions. For this patch of skin, the mean and standard deviation of the red, green, and blue values are calculated, and thresholds are set at 1, 2 and 3 standard deviations above and below the means.

The user then manually selects the area defined by the wound. The pixels in the selected wound area are then compared to the mean and standard deviation values from the sample patches of skin previously selected. If pixel values fall above or below any of the six standard deviation (+3, +2, +1, -1, -2, -3) thresholds, they are added to the mask for that standard deviation. The end result is 18 masks of the wound location that represent areas of the wound that contain abnormal color intensities.

For visualization, an image is displayed for each color channel. Using a false-color map, areas of the wound that are above the mean are highlighted red, while areas below the mean are highlighted blue. Regions of the wound that are within one standard deviation of the skin appear green, which can be seen in the output image in Figure 2. From the calculations, a numerical output is given in the form of surface area for each standard deviation threshold. The returned masks are binary, meaning there are 1's in the pixels in the segmentation and 0's outside of this area. Therefore, the sum of this mask is the total count of segmented pixels. The pixel per centimeter scale is then used to convert this pixel count to surface area. This calculation provides an area estimation for each standard deviation threshold for each color, totaling 18 area estimates. An example input and output of this program are each shown in Figure 2. Boxes are drawn over locations on the original image representing the skin sample areas and wound location area. Table 1 shows the numerical output of the program for this example. Code for these programs can be found Appendix A.

	-3	-2	-1	1	2	3
Red	1.631648	3.097223	5.451948	1.53002	0.118412	0.001326
Green	2.32193	3.77624	5.559058	3.137223	0.982182	0.029308
Blue	3.388901	4.702435	6.401372	2.014028	0.345561	0.00687

Table 1. Numerical Surface Area Output. Data from the wound shown in Figure 2. Surface area outputs for each color channel. Areas are in cm^2.

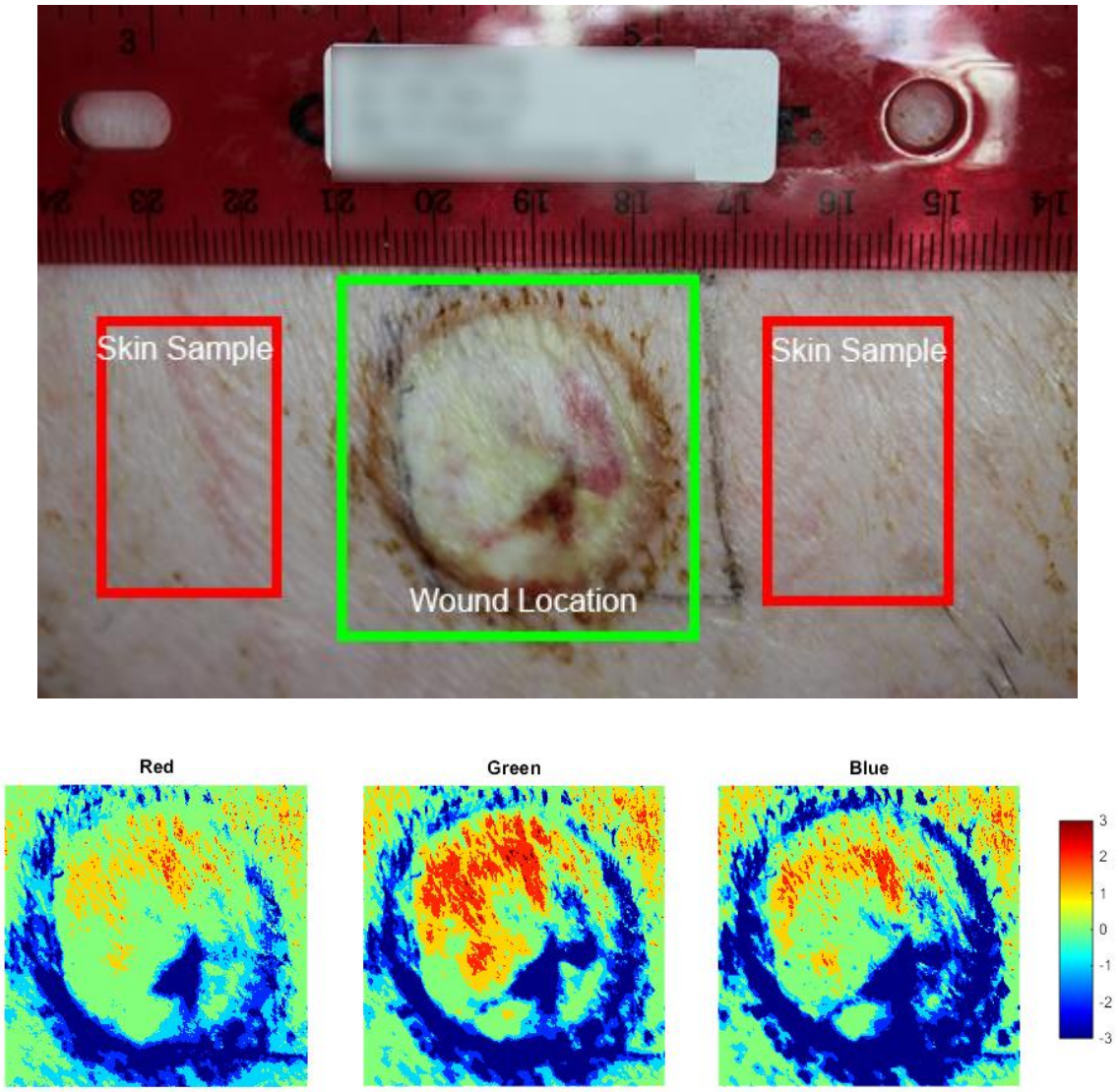


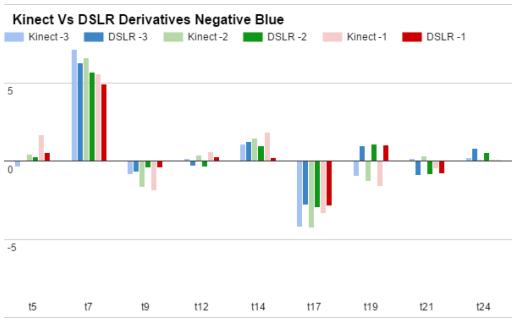
Figure 2. Input and Output of Statistical Analysis Program. The above image shows an example of an input to the Statistical Analysis Program, the red boxes highlights skin areas that were chosen for the calibration. The green box is the wound location. The bottom image is the collection of the three false-color output masks comprising the of the threshold levels for each color channel.

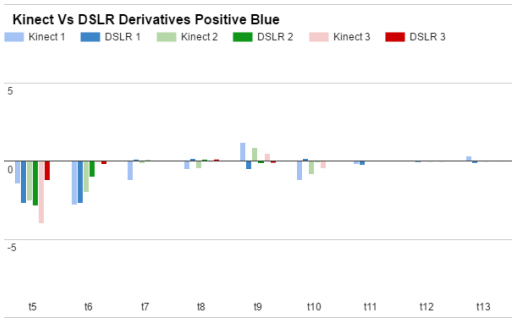
To test whether the Statistical Analysis was capable of tracking the progression of wounds, all of the images collected in the pig study were processed through this program. For the Kinect images, which contained four wounds per image, the program was modified slightly to perform the same analysis on all four wounds concurrently. Each wound in the image still used two skin samples to calculate the average skin values. The program was also modified to measure the Kinect NIR images. Unlike the three channel RGB Images, the NIR images only consists of one intensity value. For easy implication, the NIR images were given two pseudo channels consisting of only zero values. The final output is a single false-color channel mask with six surface area estimations.

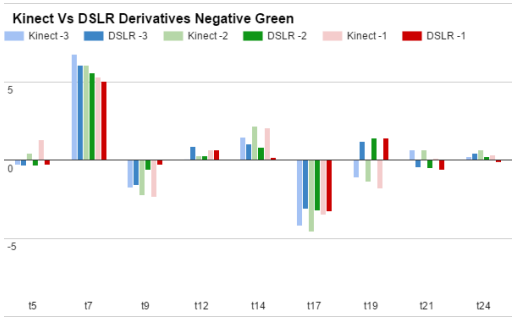
Results

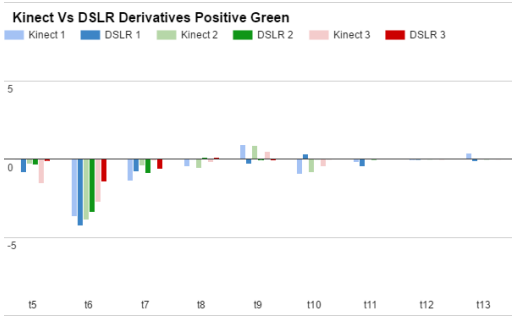
Due to the large number of outputs created by the program (18 in total), direct comparison of the wounds becomes difficult. The program's ability to calculate surface area for each color abnormality is shown in Figure 3. The surface area for a single wound's positive and negative thresholds are plotted over time for each color channel. Each graph plots the calculated DSLR and Kinect surface areas for each standard deviation threshold, in either the positive or negative for each color. Figure 4 shows the derivatives of these surface area calculations over time.

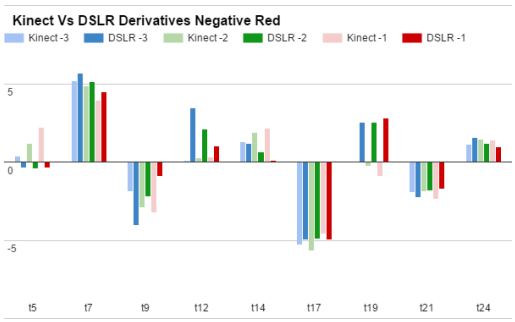
Figure 3. Surface area calculations for a single wound B3L1











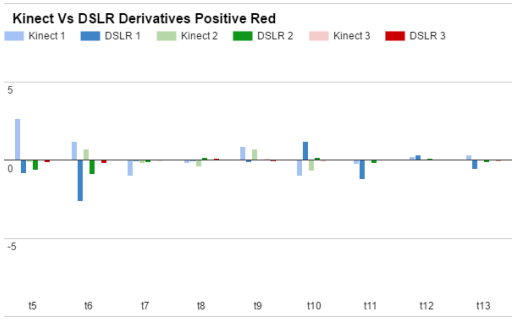


Figure 4. Derivatives of Surface Area calculations. Using values from Figure 3

Discussion

Because of the wide viewing angle of the Kinect and distance at which images were acquired, wound images obtained had significantly lower resolution as compared to the DSLR. Although the RGB Kinect camera is considered HD, the distance from the subject made the actual wound resolution significantly less. In DSLR images, the wound consists of most of the original image.

While testing the Statistical Analysis Program, several observations were made. The number of surface area outputs was large. Each wound had an output of 18 different surface area calculations. Ideally there would be one general area estimate for the wound. Adding the +1 and -1 thresholds for an individual color channel provides a reasonable estimate of all the pixels that are of abnormal values, but this is only an estimate for one color channel. None of the outputs could be considered the definitive surface area of the wound.

A second difficulty is that substantial areas of the image are incorrectly categorized. Areas of the wound that have a similar color to the skin are not picked up, but blemishes and bright areas that are unrelated to the wound are picked up. The wound output in Figure 2 displays some of these problems, as the large portions inside the wound are incorrectly labeled green. The best way to plot the output was to follow the general trend of the data, because the hard estimates were often not accurate. Accuracy of the wound mask could only be determined by visual examination, because there was no true surface area estimation to compare against. The plots in Figure 3 shows that the absolute values for the DSLR and Kinect cameras are different, but there is not a way to know which one is more accurate. However, Figure 4 shows that derivatives of the surface areas over time are showing that the

data often tends to move in the same direction, indicating that some visual change is being tracked by the program in both high resolution DSLR and lower resolution Kinect images.

Another major shortcoming arises when attempting to compare surface area calculations. Total surface area measurements give no information about the shape or location of the wound area. The Analysis program could highlight two completely separate areas of the image, but return the same surface area calculation. Due to these issues, the decision was made to abandon the Statistical Analysis program and develop a new program that would produce a single measurement, as well as attempt to diminish the effect that lighting has on the processing.

Chapter 3

Active Contour Method

An alternative approach was sought in response to the hurdles and shortcomings encountered with the statistical method. For example, to simplify the interpretation for physicians, the new method produces a single surface area estimate for a wound and determines how accurate the size and position of the area was. One failure of the statistical analysis program was that there was no true size measurement of the wound to compare to, therefore no way to tell if the program was accurate or not. To verify the accuracy of this new method, volunteers traced the outline of the wounds being analyzed. Instead of basing the new method on the previous thresholding techniques, the active contour method was chosen as the analysis method.

Background

The active contour model, or "Snakes", is an image segmentation method that is used to identify specific features in an image. The algorithm, originally published by Michael Kass in 1987 [15], works as an energy-minimizing function that adjusts a spline until the minimum energy of the points in the spline has been found. The general energy minimizing function guiding the snake is:

$$E_{snake} = E_{internal} + E_{image}$$

Equation 1. Active Contour Model

In Equation 1, E_{snake} is the energy being minimized. $E_{internal}$ is the energy of the spline itself. For points on the spline, both a high derivative and high second derivative contribute to a high internal energy. A low internal energy keeps the spline smooth and penalizes sharp movements. E_{image} is the energy of the image and can be created from a variety of properties in the image. Two commonly used image energies are pixel intensity, which is the value of the pixel, and image gradient, which are the pixel values of the derivative of the image. Pixel intensity attracts the contour to a specific color/value, while image gradient attracts the contour to edges of the image. By defining user set weights in the energy calculations, the algorithm can prioritize different image/spline features over others as shown in Equation 2.

$$E_{image} = w_1 E_{line} + w_2 E_{edge}$$

Equation 2. External energy of active contour model. Weights w_1 and w_2 adjust the effect that pixel intensity E_{line} and image gradient E_{edge} have on the contour.

The ease of use and customization properties of this algorithm has led to its prominence in image segmentation for the medical imaging field [15], [16].

The Chan-Vese model, or Active Contours without Edges, is a variation of the active contour model. This algorithm uses principles from Otsu's Method of thresholding [17], which creates a binary image by determining the point at which the histogram of pixel intensities has the minimum intra-class variance between the two created levels. The result is a binary mask where the two segmented groups have the least amount of variance within their group. The Chan-Vese model replaces the commonly used edge and intensity values of

E_{image}, and instead computes the pixel intensities distance from the average foreground (segmented portion of image) pixel intensity, and the distance from the average background (non-segmented portion of image) [18], [19]. This technique therefore creates a segmentation where the foreground and background have minimized variances of pixel intensities.

Active Contour Implementation

The active contour method uses an energy minimizing equation to deform a segmentation mask on the image. The edge of the mask is drawn to features of the image determined by the parameters set in the method. We tested two active contour methods and compared them to human traces of the wounds. The first, edge-based active contours, looks at standard image features such as intensity or derivative intensity. The second, using Chan-Vese active contours, attempts to minimize both the variance of intensities inside the segmentation and outside the segmentation. The L*a*b color space was chosen to perform the analysis. This color space was designed to better emulate human perception of color. A unique property of the space is it's separate value for lightness, which can be deprioritized, balancing out the negative effect that lighting might have on the methods.

The present implementation makes use of Matlab's built-in image analysis functions. Using the *activecontour(A,mask,method)* function, depending on the chosen *method* parameter, Matlab will run the edge or Chan-Vese active contour analysis on image *A* with the initial mask at *mask*. Implementing this built-in Matlab function, the method was designed to process a series of images from the original study data collected by GSK for the statistical analysis filter.

The present analysis uses the previous ruler line process described in the statistical method to acquire the pixel to area scale factor. The user crops the image to contain only the wound and surrounding skin. The cropped image is then displayed and the user selects sequential points around the wound to mark the boundary of the wound and skin. After the user is finished marking the wound, more points are interpolated between the user-selected points to form a spline, which is used to create a mask for the initial outline of the wound. A Gaussian blur of size [3 3] and sigma 0.5 is applied to the cropped image to reduce the noise that comes from the amount of detail present in high-resolution images and to improve segmentation [20]. The cropped, blurred RGB image is then converted to the L*a*b color space. To reduce the effect that the inconsistent lighting has on the images, only the "a" and "b" channels are averaged together to create the grayscale image for processing. Both the newly-formed grayscale image and initial mask are used by an *activecontour* function implementing the edge method and an *activecontour* function implementing the Chan-Vese method. The output from each of these methods creates two new masks which are segmentations of the wound. The cropped image, pixel centimeter conversion scale factor, initial mask, edge mask, and Chan-Vese mask are all saved to a Matlab structure array, used to keep all the data for a single wound in one location. The cropped images and masks are also saved separately outside of the structure to serve as a backup. An example of this process is shown in Figure 5. Code for these programs can be found in Appendix A.

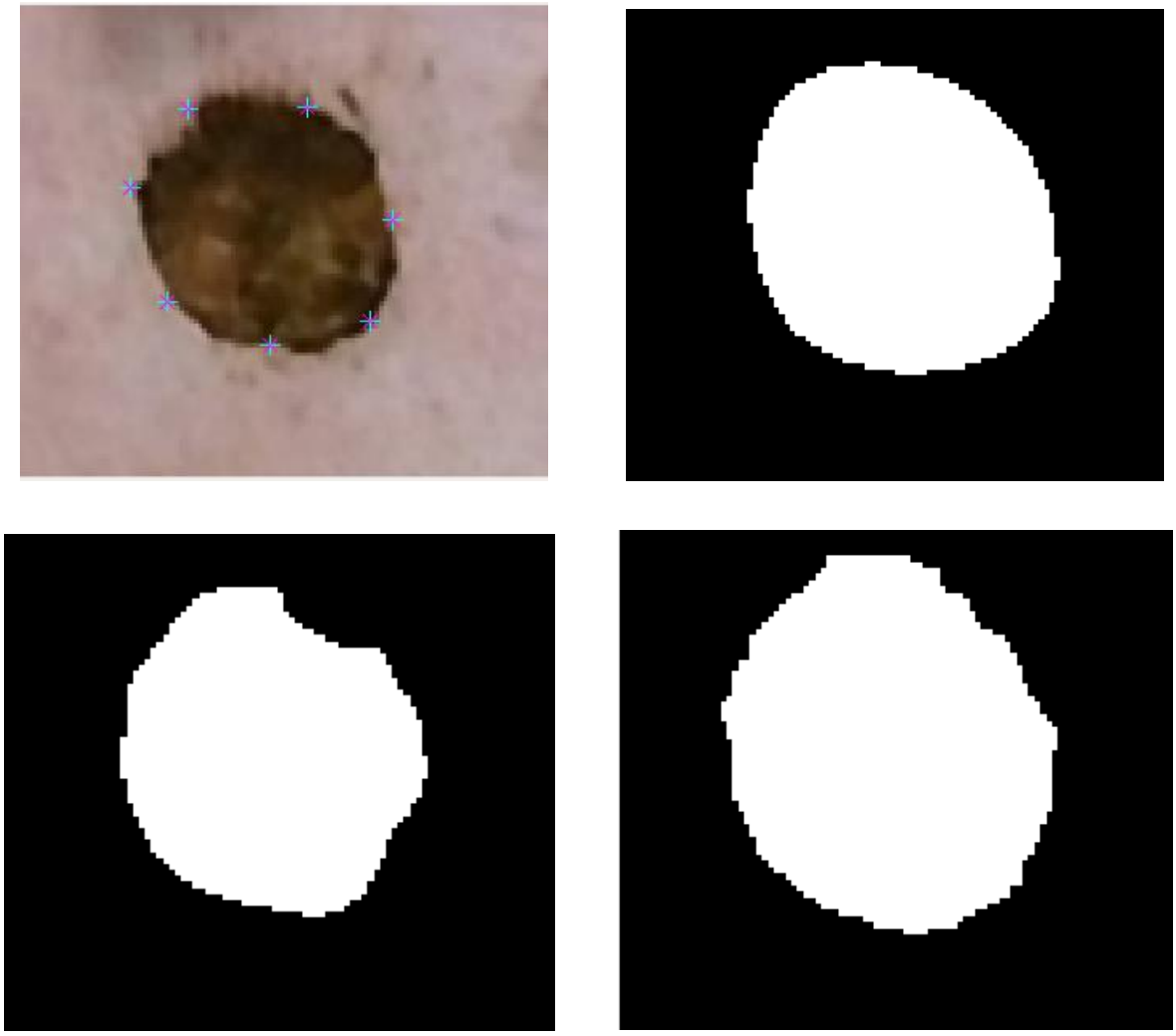


Figure 5.Masks of Active Contour Method. In clockwise order starting in upper left; cropped image with user placed boundary points, initial mask, Chan-Vese mask, edge mask.

To test the accuracy of the developed methods, a known accurate segmentation was used. To generate comparison data, 5 human volunteers digitally traced the wounds thus manually segmenting the image into wound and non-wound areas using general guidelines provided by UNC Hospitals Jaycee Burn Unit. Using the cropped images located in the structure arrays of the analyzed wounds, the *imfreehand* Matlab function was used to create

masks that are drawn by human input. Using these guidelines, instructions were created on how to trace the wound in Matlab according to the UNC Burn Unit's guidelines.

Testers are instructed to type in the ID provided by the study team so the trace can be saved with a unique name. The testers then select a wound from the folder of cropped wound images. The tester then clicks on the boundary of the wound and drags the mouse around the wound, segmenting it according to if it would be calculated in TBSA or not. After the trace has been completed, the *imfreehand* segment is converted to a mask and is overlaid on top of the cropped image, highlighting the area of segmentation. The tester is prompted to Continue, Quit, or Redo the current trace if they believe an error was made. The mask is then added to the structure array of the wound.

To gauge the accuracy of the segmentations, Dice's Coefficient was used to compare size and location of the masks. Dice's Coefficient is commonly used to compare items of a set [21]. It is calculated using the following equation.

$$QS = \frac{2|A \cap B|}{|A| + |B|}$$

Equation 3. Dice's Coefficient

In this equation A and B are masks and the intercept of the two masks is the number of pixels that are segmented in both masks. QS is the coefficient value and is between 0 and 1. If two items are exactly the same, double the union is equal to their total sum and the coefficient is 1. This coefficient is used to validate the calculated surface area and to insure that the surface areas consist for the same regions. For a single wound, each mask's Dice's coefficient is evaluated against every other mask (excluding the initial mask). To calculate surface area, the total number of segmented pixels in a mask is divided by the pixel per centimeter

conversion factor. The outputs of this Active Contours Method include the structure array, the dice coefficients, and the surface area measurements.

As a test case, a single wound, L2, from each pig was used as the seed for comparing active contours to human tracing. A total of 270 images of wound L2 were analyzed by both active contours and human tracing. Images from both the DSLR and Kinect were analyzed. To measure Dice's coefficient, images must be the same resolution, so the output masks from the two cameras could not be compared directly. However, the outputs from active contours were compared to the human traces to determine accuracy.

Results

While compiling the data, any images the testers may have accidently skipped were giving NaN values. To compare accuracy of the Active Contour Methods, Dice's coefficient values can be compared. A value of 0.7 or greater is generally considered to indicate good agreement [22], although the value itself is considered only useful for comparing methods of segmentations for the same object. Using box plots, the variance between the individual testers can be compared to the variance of the Active Contour Methods and the testers. For these plots, tester masks were only compared to other tester masks. Active Contour masks were not compared with each other. The first plot shows all of the Dice's Coefficients for the DSLR images, while the second shows all Dice's Coefficients for Kinect images, shown in Figure 6.

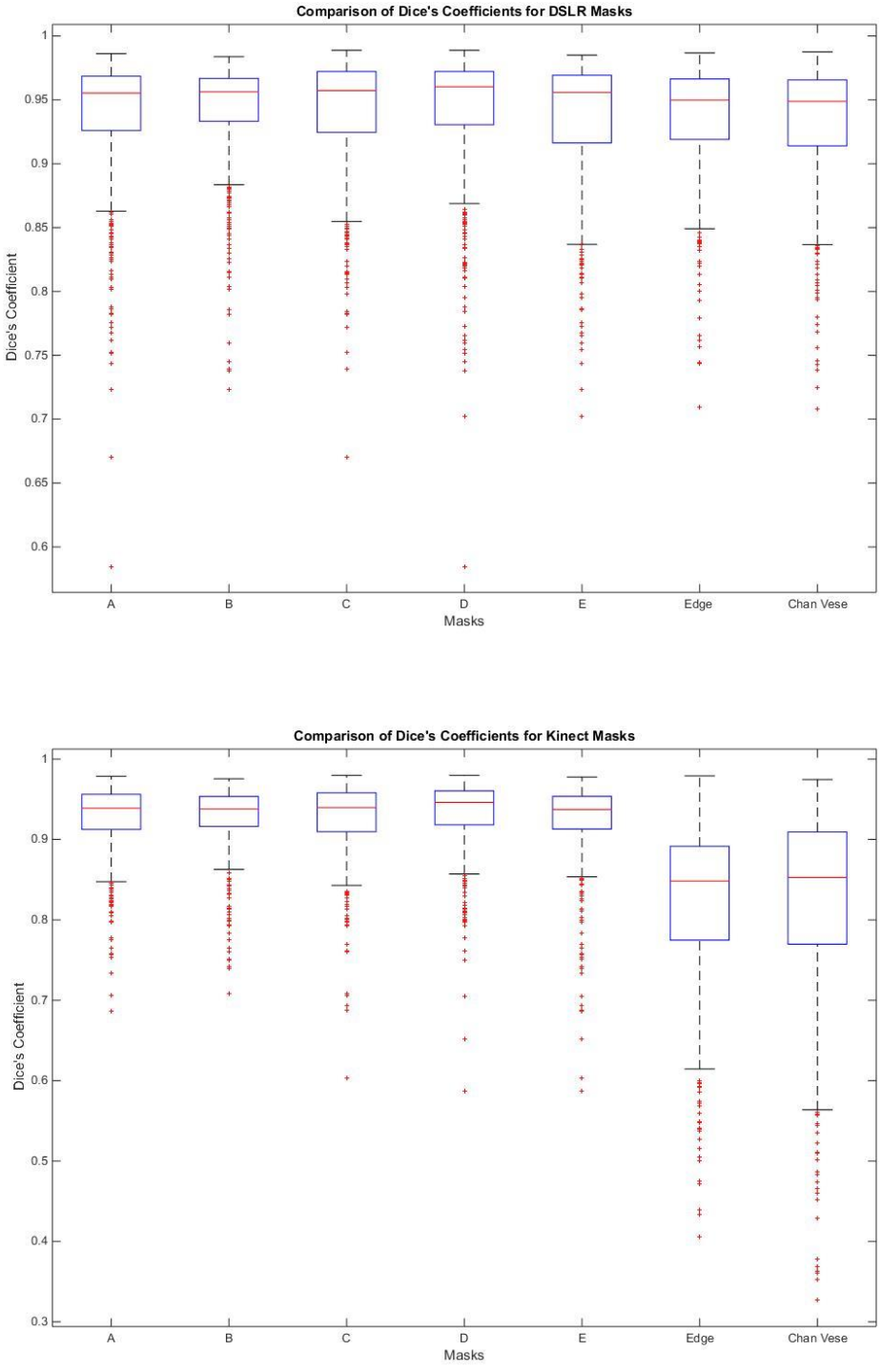


Figure 6. Mask variance comparisons using Dice's Coefficient.

The Active Contour Methods resulted in Dice's Coefficients that were above the 0.7 threshold for agreement. For n=710 comparisons between DSLR images, the edge method's Dice's Coefficient mean, reported as mean ± standard deviation, was 0.9399 ±0.0397 with a range of 0.7095 to 0.9868. The Chan-Vese Dice's Coefficient mean was 0.9362 ±0.0435 with a range of 0.708 to 0.9876. The Kinect Images did show a larger difference in variance of Dice's Coefficient n=670 comparisons. For the edge method, the Dice's Coefficient mean was 0.8154 ±0.1014 with a range of 0.4051 to 0.9792 and for the Chan Vese method the Dice's Coefficient mean was 0.8186 ±0.1259 with a range of 0.3268 to 0.9746. These results agree with a visual examination of the results of the test. The DSLR outputs visually appear to match the wound boundaries than in the Kinect outputs. Examples of outputs for both DSLR and Kinect are shown in Figure 7.

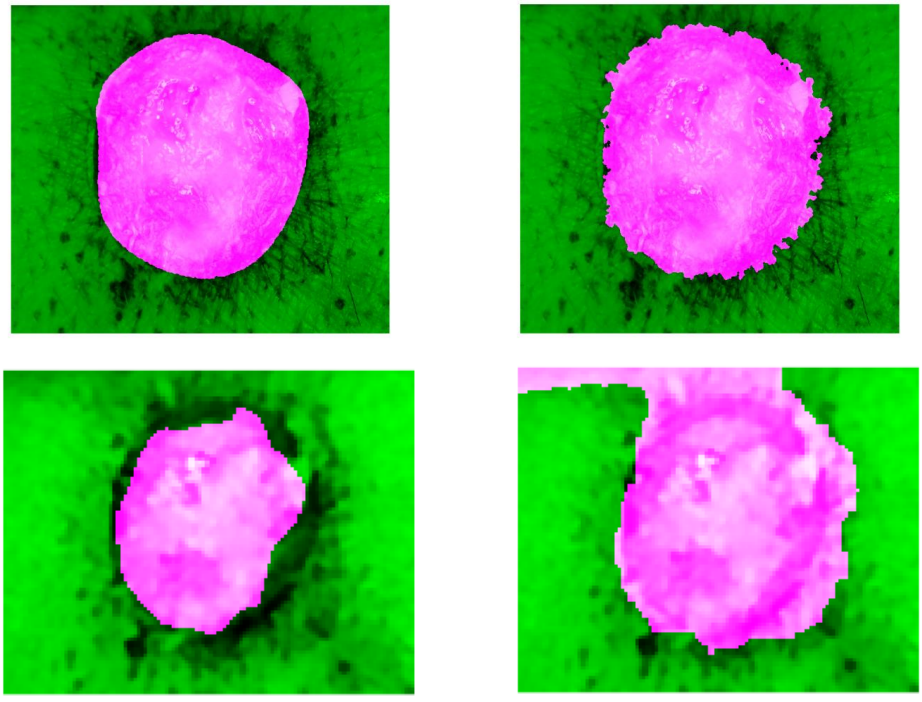


Figure 7 Active Contour Outputs. From Pig A3 Wound L2 Day 16; Top is DSLR, Bottom is Kinect, Left side is edge method, Right side is Chan Vese method. Purple represents areas of the image segmented as a wound. Green is the non-wound area.

Plotting the surface area calculations for a specific wound over time attempts to show the progression of the wound size over the course of the study, as shown in Figure 8. The dotted lines in the plot represent the tester's traces and the solid lines represent the Active Contour Methods. Appendix B has a collection of these plots for all pigs.

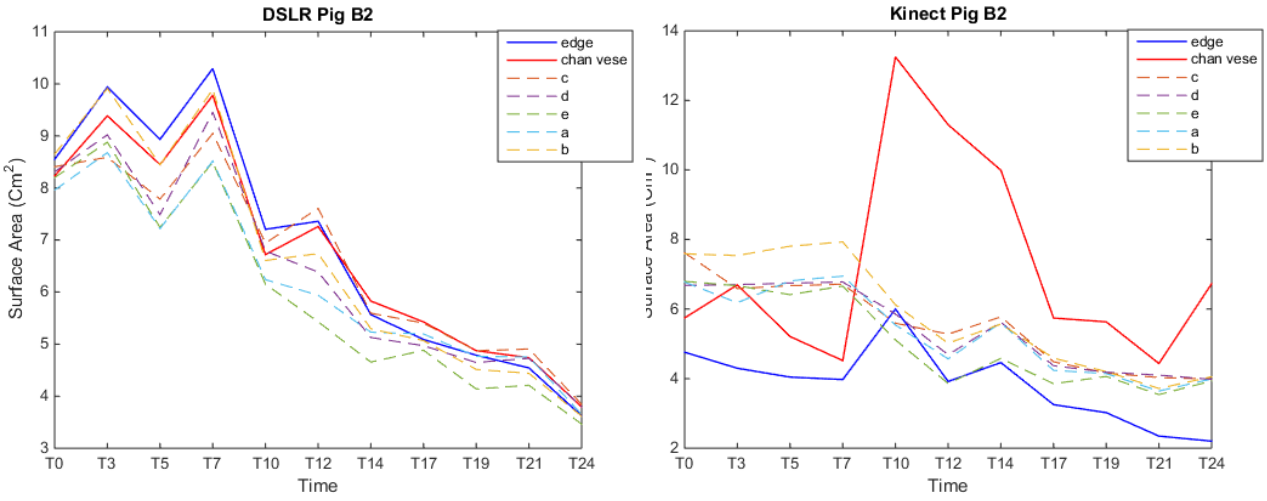


Figure 8. DSLR and Kinect Surface Areas over Time

The overall trends of these surface areas can be viewed by looking at all DSLR and Kinect surfaces areas plotted contiguously. The DSLR data fits well with the testers' traces, with a slight tendency to overestimate. The Kinect data is often significant different than the testers' traces. This is shown in Figure 9.

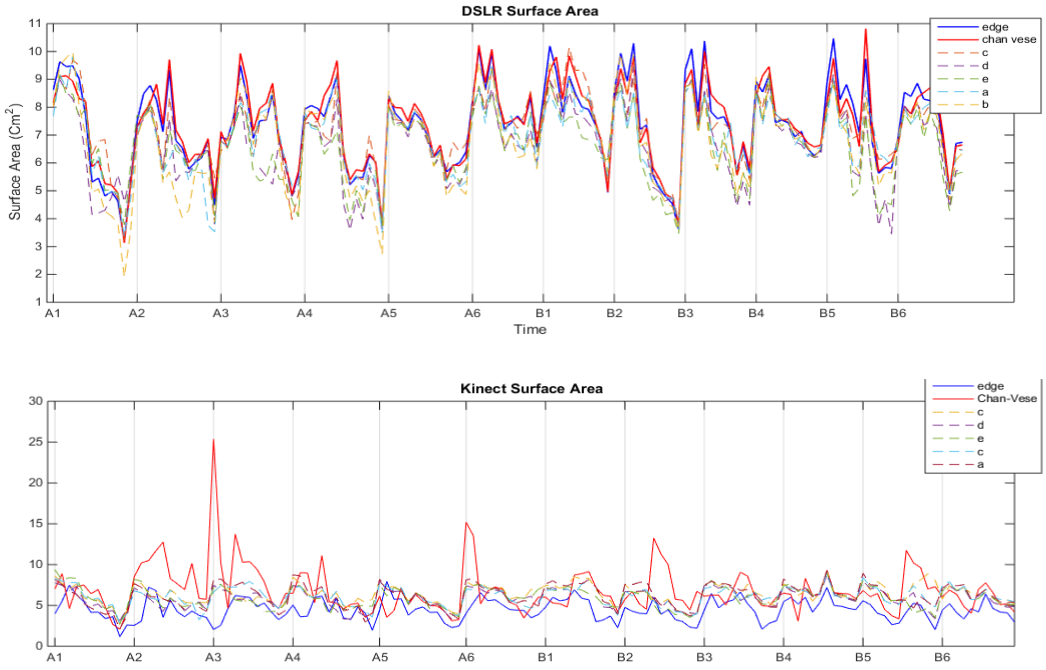


Figure 9. Top, All DSLR surface area calculations; Bottom, All Kinect surface area calculations. Red and Blue lines represent edge and Chan Vese surface areas. Dotted lines represent the Trace's surface areas.

Discussion

The second attempt at segmenting the burn wounds proved much more successful than the first. For the DSLR images, both the edge and Chan-Vese method's had agreeable Dice's coefficients with the traces, meaning this current method is capable of providing segmentations that are comparable to a human. Although, it is possible that these accurate results are coming from a beneficial initial mask rather than the Active Contour Methods themselves. In the Kinect images, the output was capable of producing large changes from the initial mask. The DSLR images did not have as significant changes to the initial mask, especially in the edge method. This could be due to a variety of factors, the Gaussian blur on the images could not be strong enough to counteract fine details present in the high resolution

images, or the Active Contour Methods could be going through too few iterations relative to the number of pixels.

A noticeable difference between the Kinect and DSLR analysis is the run time. Due to the large number of pixels in the DSLR images, the run time was around 90 seconds versus the Kinect images runtime of 3 seconds.

The output from the Kinect had large amounts of variance in the Dice Coefficient values. This means that the masks from the outputs did segment the same areas of the traces. The lowest non-outlier data in both Edge and Chan Vese methods were below the 0.7 threshold for good agreement between sets. This does not mean the Kinect is not capable of taking good images of wounds, but the method used in this study did not provide images that were high enough quality for analysis.

Chapter 4

Conclusions

While the statistical analysis program failed to provide reasonable results, it did provide insights into further development of dermal wound analysis. Testing could be redone of the statistical analysis program and compared to tester traces to determine compare it to the active contours method, but the issue of deriving one mask to compare from the series of 18 generated still remains. One advantage this analysis method has over Active Contours is that it can represent changes in color. This could prove useful if future progress in this field requires classification of pixels. Knowing relative pixel intensity to the skin could be a possible classification feature if a method were to be designed for classifying areas of the wound, or severity of the wound.

The Active Contour Method has shown agreeable surface area calculations for high-resolution images, but it is unknown how much of an effect the initial mask has on the process. To test effect of initial mask, simplifying the initial mask to a circle, or even the border of the cropped image itself should be performed. By using a less favorable initial mask, the method will need to rely more on the parameters of the active contours functions. Parameters of the *activecontours* method can also be adjusted that control the smoothing factor of the contour, as well as its tendency to grow inwards or outwards. The latter would be useful if the initial mask is either entirely surrounding or enveloped in the wound. Further, the number of iterations can also be changed. A larger iteration number provides the contour

with opportunity to move across more pixels, which would help with the larger resolution images, but this comes with an increased run time. The weight of the L*a*b channels can also be adjusted easily in this method. Fine-tuning L*a*b weights to increase the contrast of the wounds could provide a more accurate segmentation.

All of images used in these experiments were uniform wounds of similar severity. Burns are not traditionally perfect circles. These methods still need testing on actual burn patients. Due to the limitations of 2d photography, only wounds that can be mostly contained in a single image will be able to receive surface area estimates. It would be difficult to compound analysis of multiple images of the same wound from different angle, for example a burn wrapping around an arm. Imaging burn wounds on real patients would increase the knowledge of the capabilities and limits of these methods.

Although the Kinect data used in the study was poor, current research is being done on adjusting the Kinect for use in a 'near mode'. The near mode would allow for close NIR/Depth images of the burns that could be analyzed. Receiving an accurate depth value for pixels would enable a more accurate surface area calculation by using the third dimension. We still recommend segmenting the initial wound with the color images, as they are the highest resolution and have the highest variance of pixel intensities.

In conclusion, the Chan-Vese and edge based Active Contour Methods have thus far proven an effective way to measure the surface area of burn wounds in high resolution images, although this may be due to a favorable starting location. Both of these methods generated masks with agreeable Dice's Coefficients between human testers for the high-resolution images. The surface area calculations from these methods were able to track the

size of the wound over time. The development and testing of these methods is the first step in designing a system capable of properly calculating burn wound surface area measurements.

Due to limitations of the data received and the Kinect 2.0 itself, there was not enough comparable data to do a scientific comparison between the Kinect 2.0 and the standard DSLR camera. Further testing is required before the Kinect can be properly assessed as a tool for dermatological imaging.

APPENDIX A: Code

```
function [outputc,outpute] = ACmain(oimage,mask)
%Runs the Chan Vese and Edge Active Contour Method
%Apply a blur
h = fspecial('gaussian');
image=imfilter(oimage,h);
%Convert to LAB
lab_img=rgb2lab(image);
gray=rgb2gray(image);
img_a=lab_img(:,:,2);
img_b=lab_img(:,:,3);
img_ab=img_a+img_b;

%Run method and perform closing
coutput=activecontour(img_ab,mask,'chan-vese');
eoutput=activecontour(img_ab,mask,'edge');
se = strel('disk',3);
outputc = imclose(coutput,se);
outpute = imclose(eoutput,se);

%Plot Both Output Masks Over Image
subplot(1,3,1);
imshowpair(image,outputc);
subplot(1,3,2);
imshowpair(image,outpute);
subplot(1,3,3);
imshow(image);



function output = dice(mask1, mask2)
%Perform Dice's Coefficient Analysis on Two Masks
%Checks to make sure masks are the same size
if size(mask1)~= size(mask2)
   output=NaN;
else
output= 2*nnz(mask1&mask2)/(nnz(mask1) + nnz(mask2));
end



function output = dslr_ac (i)
%Performm AC Analysis on DSLR images
imshow(i);
%Measure One Inch on Image, Calculates Pixels per cm
distline = imdistline;
pause;
cm = getDistance(distline)*.3937;
%Crop Image to just cointain wound
cimage = imcrop(i);
clf;
%Create Initial Mask
m = mask(cimage);
%Run AC Methods
[oc,oe]=ACmain(cimage,m);
```

```
%Calculate Surface Areas
sc=sum(sum(oc))/(cm^2);
se=sum(sum(oe))/(cm^2);
%Save Outputs in Structure
output=struct('cropped',cimage,'mask',m,'scale',cm,'chan_vese_mask',oc,'edge_mask',oe,'chan_vese_sa',sc,'edge
_sa',se);


function output=dslr_ac_batch()
%Run Batch Analysis using Active Contours
running = 1;
%Data Location
p = 'D:\DSLR\DSLR\';
while running
    %Load File
   [f,p]=uigetfile(strcat(p,'*.jpg'));
   image = imread([p f]);
   filename = (inputdlg('Whats the name?'));
   name = filename{1};
   output = dlsr_ac(image);



   %Saving the Output
   %Folders must exist before running program
   imwrite(output.cropped,sprintf(['/Data/cropped/cropped_' name '.png']));
   imwrite(output.mask,sprintf(['/Data/initialmask_' name '.png']));
   imwrite(output.chan_vese_mask,sprintf(['/Data/chanveseoutput_' name '.png']));
   imwrite(output.edge_mask,sprintf(['/Data/edgeoutput_' name '.png']));
   eval([name '= output']);
   save(sprintf(['/Data/batch/' name '.mat']),name);

   %UI To Keep Going
   button = questdlg('Keep Going?',' ','Yes','No','Yes');
   if strcmp('Yes',button);
      running = 1;
      close gcf;
   else
      running = 0;
   end
end
end
function output = dslr_sa(image)
%Run SA Analysis on DSLR Images
%Crop Skin Samples and Wound
skin1 = imcrop(image);
skin2 = imcrop(image);
w1   = imcrop(image);
%Measure One inch in pixels, convert to cm
distline = imdistline;

pause;
meter = getDistance(distline)*.3937;
dist = meter;
```

```
%Input Name
filename = (inputdlg('Whats the name?'));

name = filename{1};
close;


%Seperate Color Channels
rskin1 = skin1(:,:,1);
figure

gskin1 = skin1(:,:,2);
bskin1 = skin1(:,:,3);
rskin2 = skin2(:,:,1);
gskin2 = skin2(:,:,2);
bskin2 = skin2(:,:,3);

%Combine Skin Samples
rskin = vertcat(reshape(rskin1,[],1),reshape(rskin2,[],1));
gskin = vertcat(reshape(gskin1,[],1),reshape(gskin2,[],1));
bskin = vertcat(reshape(bskin1,[],1),reshape(bskin2,[],1));


%Take Mean and STD of Skin Samples
rmean = mean2(rskin);
gmean = mean2(gskin);
bmean = mean2(bskin);
rstd  = std2(rskin);
gstd  = std2(gskin);
bstd  = std2(bskin);

%Create Output Array
o1 = zeros(size(w1,1),size(w1,2),3,6);

%Cycle Through Pixels in image, compare to postivive STD Tresholds then negative,
%.. if True set Output Array value to 1 (or True)
for x = 1:size(w1,1)
   for y = 1:size(w1,2)
      for z = 1:3
         if w1(x,y,1) > (rmean + z*rstd)
            o1(x,y,1,z+3) = 1;
         end
         if w1(x,y,2) > (gmean + z*gstd)
            o1(x,y,2,z+3) = 1;
         end
         if w1(x,y,3) > (bmean + z*bstd)
            o1(x,y,3,z+3) = 1;
         end
      end
      for z = -3:-1
         if w1(x,y,1) < (rmean + z*rstd)
            o1(x,y,1,z+4) = 1;
         end
         if w1(x,y,2) < (gmean + z*gstd)
```

```
            o1(x,y,2,z+4) = 1;
        end
        if w1(x,y,3) < (bmean + z*bstd)
            o1(x,y,3,z+4) = 1;
        end
    end
  end
end



%Create Array and Calcuate Surface Areas for Each STD Threshold
  n1 = zeros(3,6);
  for d = 1:6
    for c = 1:3
      n1(c,d) = (sum(sum(o1(:,:,c,d))/(dist^2)));
    end
  end


%Create Combined Output Display
c1 = zeros(size(w1,1),size(w1,2),3);

%Combine the Six Output Masks into one mask per color channel
for z = 1:3
  for x = 1: size(w1,1)
    for y = 1: size(w1,2)
      if     o1(x,y,z,1) == 1
             c1(x,y,z)   = -3;
      elseif  o1(x,y,z,2) == 1
             c1(x,y,z)   = -2;
      elseif  o1(x,y,z,3) == 1
             c1(x,y,z)   = -1;
      elseif  o1(x,y,z,6) == 1
             c1(x,y,z)   = 3;
      elseif  o1(x,y,z,5) == 1
             c1(x,y,z)   = 2;
      elseif  o1(x,y,z,4) == 1
             c1(x,y,z)   = 1;
      end
    end
  end
  %Display Three Color Channel Output Masks
  subplot(1,3,z);
  imshow(c1(:,:,z));
  colormap jet;
  caxis([-3,3]);


end

subplot(1,3,1);
title('Red');
subplot(1,3,2);
```

```
title('Green');
subplot(1,3,3);
title('Blue');
%Save Figure
file = sprintf(['/Data/dslr/png/' name '.png']);
saveas(gcf,file);
%Save Surface Areas as .xls
xlswrite(['C:\Data\dslr\xls\' name '.xlsx'],n1(:,:));
%Save data as Structure
output=struct('CmPixels',dist, 'Cropped', w1, 'RedMean', rmean, 'RedSTD', rstd, 'GreenMean', gmean,
'GreenSTD', gstd, ...
   'BlueMean', bmean, 'BlueSTD', bstd, 'OutputImage', c1, 'OutputArea', n1 );
eval([name '= output']);
save(sprintf(['/Data/pig/test/batch/' name '.mat']),name);

end

function dslr_sa_batch()
%Program to Help Run Batch Analysis using Statistical Analysis Filter
running = 1;
%Location of Data
p = 'D:\DSLR\DSLR\';
while running
   %Load Image
   [f,p]=uigetfile(strcat(p,'*.png'));
   image = imread([p f]);
   %Perform Analysis
   skincalc4(image);
   %UI Ask to Continue
   button = questdlg('Keep Going?',' ','Yes','No','Yes');
   if strcmp('Yes',button);
      running = 1;
      close gcf;
   else
      running = 0;
   end
end
end


function output = freehand(image)
%Creating the Manual Freehand Mask
imshow(image)
set(gcf,'position',[200 100 850 750]);
H=imfreehand;
output=H.createMask();
close;
imshowpair(image,output);


function freehand_batch()
%Creates Freehand Trace Input
tester = inputdlg('Which Tester?');
letter=char(tester);
```

```
running = 1;
repeat = 0;
%Location of Cropped Images
p = 'C:\Data\cropped\';
while running
   if repeat == 0;
      %Load Image
      [f,p]=uigetfile(strcat(p,'*.png'));
      image = imread([p f]);
      %Extract Image Name
      name=f(9:end-4);
   end
   output = freehand(image);



   %Load Output Strcuture with Same Name as image
   mat = load(sprintf(['/Data/Test/' name]));
   %Save Trace in Structure and Save Structure
   mat.(name).(letter)=output;
   mat2=mat.(name);
   eval([name '= mat2']);
   save(sprintf(['/Data/test/' name '.mat']),name);


   %Save Trace Image
   imwrite(output,sprintf(['/Data/traces/' letter '/' letter 'freehand_' name '.png']));

   repeat=0;
   %Display Trace
   disp(f);

   %UI to Keep Going or Redo
   button = questdlg('Keep Going?',' ','Yes','No','Repeat','Yes');
   if strcmp('Yes',button);
      running = 1;
      close gcf;
   elseif strcmp('Repeat',button);
      running = 1;
      repeat = 1;

   else
      running = 0;
   end
end
end
```

```
function diceval = gatherdata(diceval)
%Used to Compile Mask Dice Data from Structures
addpath('C:\Data\batch')

files = dir('C:\Data\batch');
files=files';
i=1;
for file=files(5:end)

    filename = file.name;
    temp = load(filename);
    name=char(fieldnames(temp));
    disp(name);

    eval(['current=temp.' name ';']);
    diceval(1,1,i)= dice(current.a,current.f);
    diceval(1,2,i)= dice(current.a,current.c);
    diceval(1,3,i)= dice(current.a,current.d);
    diceval(1,4,i)= dice(current.a,current.e);
    diceval(1,5,i)= dice(current.a,current.edge_mask);
    diceval(1,6,i)= dice(current.a,current.chan_vese_mask);
    diceval(2,2,i)= dice(current.f,current.c);
    diceval(2,3,i)= dice(current.f,current.d);
    diceval(2,4,i)= dice(current.f,current.e);
    diceval(2,5,i)= dice(current.f,current.edge_mask);
    diceval(2,6,i)= dice(current.f,current.chan_vese_mask);
    diceval(3,3,i)= dice(current.c,current.d);
    diceval(3,4,i)= dice(current.c,current.e);
    diceval(3,5,i)= dice(current.c,current.edge_mask);
    diceval(3,6,i)= dice(current.c,current.chan_vese_mask);
    diceval(4,4,i)= dice(current.d,current.e);
    diceval(4,5,i)= dice(current.d,current.edge_mask);
    diceval(4,6,i)= dice(current.d,current.chan_vese_mask);
    diceval(5,5,i)= dice(current.e,current.edge_mask);
    diceval(5,6,i)= dice(current.e,current.chan_vese_mask);
    diceval(6,6,i)= dice(current.edge_mask,current.chan_vese_mask);
    i=i+1;




end


function output = gatherdata2(sa)
%Used to Compile Surface Area data from Structures
addpath('C:\Data\batch')

files = dir('C:\Data\batch');
files=files';
i=1;
for file=files(5:end)

    filename = file.name;
```

```
        temp = load(filename);
        name=char(fieldnames(temp));
        disp(name);

        eval(['current=temp.' name ';']);
        sa(1,i)=sum(sum(current.a))/(current.scale)^2;
        sa(2,i)=sum(sum(current.f))/(current.scale)^2;
        sa(3,i)=sum(sum(current.c))/(current.scale)^2;
        sa(4,i)=sum(sum(current.d))/(current.scale)^2;
        sa(5,i)=sum(sum(current.e))/(current.scale)^2;
        sa(6,i)=current.chan_vese_sa;
        sa(7,i)=current.edge_sa;
        i=i+1;
end
output=sa;
end
```

```
function output = kinect_ac (i)
%Perform AC Analysis on Kinect Images
imshow(i);
%Record Lenght of 30 cm (Length of an Entire Ruler) calculate pixels per cm
distline = imdistline;
pause;
cm = getDistance(distline)/30;
%Crop Image
cimage = imcrop(i);
clf;
%Create Initial Mask
m = mask(cimage);
%Run Analysis
[oc,oe]=ACmain(cimage,m);
sc=sum(sum(oc))/(cm^2);
se=sum(sum(oe))/(cm^2);
%Output Data as Structure
output=struct('cropped',cimage,'mask',m,'scale',cm,'chan_vese_mask',oc,'edge_mask',oe,'chan_vese_sa',sc,'edge
_sa',se);


function output=kinect_ac_batch()
running = 1;
p = 'C:\Users\Peter\Desktop\7_7_16 Data\';
while running

    [f,p]=uigetfile(strcat(p,'*.png'));
    mirrorimage = imread([p f]);
    image = flip(mirrorimage ,2);
    filename = (inputdlg('Whats the name?'));
    name = filename{1};
    output = kdac(image);



    %Saving the Output

    imwrite(output.cropped,sprintf(['/Users/Peter/Data/cropped/cropped_' name '.png']));
    imwrite(output.mask,sprintf(['/Users/Peter/Data/initialmask_' name '.png']));
    imwrite(output.chan_vese_mask,sprintf(['/Users/Peter/Data/chanveseoutput_' name '.png']));
    imwrite(output.edge_mask,sprintf(['/Users/Peter/Data/edgeoutput_' name '.png']));

    %%f2=f(1:end-4);
    eval([name '= output']);
    save(sprintf(['/Users/Peter/Data/batch/' name '.mat']),name);


    button = questdlg('Keep Going?',' ','Yes','No','Yes');
    if strcmp('Yes',button);
        running = 1;
        close gcf;
    else
        running = 0;
    end
```

```
end
end



function output = kinect_sa(image)
%Perform Statistical Analysis on Kinect Images
%Select Four Wounds from Image and Four Skin Samples
disp('skin1');
skin1 = imcrop(image);
disp('w1');
w1   = imcrop(image);

disp('skin2');
skin2 = imcrop(image);
disp('w2');
w2= imcrop(image);

disp('skin3');
skin3 = imcrop(image);
disp('w3');
w3   = imcrop(image);

disp('skin4');
skin4 = imcrop(image);
disp('w4');
w4   = imcrop(image);

%Measure Lenght of 30 cm (Lenght of Ruler) and convert to pixels per cm
distline = imdistline;
pause;
ruler = getDistance(distline);
filename = (inputdlg('Whats the name?'));
name = filename{1};
dist = ruler/30;
close;
%Calculate Skin Means and STD
rskin1 = skin1(:,:,1);
gskin1 = skin1(:,:,2);
bskin1 = skin1(:,:,3);
rmean1 = mean2(rskin1);
gmean1 = mean2(gskin1);
bmean1 = mean2(bskin1);
rstd1  = std2(rskin1);
gstd1  = std2(gskin1);
bstd1  = std2(bskin1);

rskin2 = skin2(:,:,1);
gskin2 = skin2(:,:,2);
bskin2 = skin2(:,:,3);
rmean2 = mean2(rskin2);
gmean2 = mean2(gskin2);
bmean2 = mean2(bskin2);
rstd2  = std2(rskin2);
gstd2  = std2(gskin2);
```

```
bstd2  = std2(bskin2);

rskin3 = skin3(:,:,1);
gskin3 = skin3(:,:,2);
bskin3 = skin3(:,:,3);
rmean3 = mean2(rskin3);
gmean3 = mean2(gskin3);
bmean3 = mean2(bskin3);
rstd3  = std2(rskin3);
gstd3  = std2(gskin3);
bstd3  = std2(bskin3);

rskin4 = skin4(:,:,1);
gskin4 = skin4(:,:,2);
bskin4 = skin4(:,:,3);
rmean4 = mean2(rskin4);
gmean4 = mean2(gskin4);
bmean4 = mean2(bskin4);
rstd4  = std2(rskin4);
gstd4  = std2(gskin4);
bstd4  = std2(bskin4);




%Create Output Arrays
o1 = zeros(size(w1,1),size(w1,2),3,6);
o2 = zeros(size(w2,1),size(w2,2),3,6);
o3 = zeros(size(w3,1),size(w3,2),3,6);
o4 = zeros(size(w4,1),size(w4,2),3,6);

%Cycle Through Pixels in image, compare to postivive STD Tresholds then negative,
%.. if True set Output Array value to 1 (or True)
for x = 1:size(w1,1)
   for y = 1:size(w1,2)
      for z = 1:3
         if w1(x,y,1) > (rmean1 + z*rstd1)
            o1(x,y,1,z+3) = 1;
         end
         if w1(x,y,2) > (gmean1 + z*gstd1)
            o1(x,y,2,z+3) = 1;
         end
         if w1(x,y,3) > (bmean1 + z*bstd1)
            o1(x,y,3,z+3) = 1;
         end
      end
      for z = -3:-1
         if w1(x,y,1) < (rmean1 + z*rstd1)
            o1(x,y,1,z+4) = 1;
         end
         if w1(x,y,2) < (gmean1 + z*gstd1)
            o1(x,y,2,z+4) = 1;
         end
         if w1(x,y,3) < (bmean1 + z*bstd1)
```

```
            o1(x,y,3,z+4) = 1;
        end
    end
  end
end


for x = 1:size(w2,1)
  for y = 1:size(w2,2)
    for z = 1:3
      if w2(x,y,1) > (rmean2 + z*rstd2)
        o2(x,y,1,z+3) = 1;
      end
      if w2(x,y,2) > (gmean2 + z*gstd2)
        o2(x,y,2,z+3) = 1;
      end
      if w2(x,y,3) > (bmean2 + z*bstd2)
        o2(x,y,3,z+3) = 1;
      end
    end
    for z = -3:-1
      if w2(x,y,1) < (rmean2 + z*rstd2)
        o2(x,y,1,z+4) = 1;
      end
      if w2(x,y,2) < (gmean2 + z*gstd2)
        o2(x,y,2,z+4) = 1;
      end
      if w2(x,y,3) < (bmean2 + z*bstd2)
        o2(x,y,3,z+4) = 1;
      end
    end
  end
end

for x = 1:size(w3,1)
  for y = 1:size(w3,2)
    for z = 1:3
      if w3(x,y,1) > (rmean3 + z*rstd3)
        o3(x,y,1,z+3) = 1;
      end
      if w3(x,y,2) > (gmean3 + z*gstd3)
        o3(x,y,2,z+3) = 1;
      end
      if w3(x,y,3) > (bmean3 + z*bstd3)
        o3(x,y,3,z+3) = 1;
      end
    end
    for z = -3:-1
      if w3(x,y,1) < (rmean3 + z*rstd3)
        o3(x,y,1,z+4) = 1;
      end
      if w3(x,y,2) < (gmean3 + z*gstd3)
        o3(x,y,2,z+4) = 1;
      end
      if w3(x,y,3) < (bmean3 + z*bstd3)
```

```
            o3(x,y,3,z+4) = 1;
        end
    end
  end
end

for x = 1:size(w4,1)
  for y = 1:size(w4,2)
    for z = 1:3
      if w4(x,y,1) > (rmean4 + z*rstd4)
        o4(x,y,1,z+3) = 1;
      end
      if w4(x,y,2) > (gmean4 + z*gstd4)
        o4(x,y,2,z+3) = 1;
      end
      if w4(x,y,3) > (bmean4 + z*bstd4)
        o4(x,y,3,z+3) = 1;
      end
    end
    for z = -3:-1
      if w4(x,y,1) < (rmean4 + z*rstd4)
        o4(x,y,1,z+4) = 1;
      end
      if w4(x,y,2) < (gmean4 + z*gstd4)
        o4(x,y,2,z+4) = 1;
      end
      if w4(x,y,3) < (bmean4 + z*bstd4)
        o4(x,y,3,z+4) = 1;
      end
    end
  end
end


%Create Array and Calcuate Surface Areas for Each STD Threshold
  n1 = zeros(3,6,4);
  for d = 1:6
    for c = 1:3
      n1(c,d,1) = (sum(sum(o1(:,:,c,d)))/(dist^2));
    end
  end

  for d = 1:6
    for c = 1:3
      n1(c,d,2) = (sum(sum(o2(:,:,c,d)))/(dist^2));
    end
  end



  for d = 1:6
    for c = 1:3
      n1(c,d,3) = (sum(sum(o3(:,:,c,d)))/(dist^2));
    end
  end
```

```
    for d = 1:6
        for c = 1:3
            n1(c,d,4) = (sum(sum(o4(:,:,c,d)))/(dist^2));
        end
    end


%Create Combined Output Display
c1 = zeros(size(w1,1),size(w1,2),3);
c2 = zeros(size(w2,1),size(w2,2),3);
c3 = zeros(size(w3,1),size(w3,2),3);
c4 = zeros(size(w4,1),size(w4,2),3);

%Combine the Six Output Masks into one mask per color channel
    %Display Three Color Channel Output Masks

for z = 1:3
    for x = 1: size(w1,1)
        for y = 1: size(w1,2)
            if      o1(x,y,z,1) == 1
                    c1(x,y,z)   = -3;
            elseif  o1(x,y,z,2) == 1
                    c1(x,y,z)   = -2;
            elseif  o1(x,y,z,3) == 1
                    c1(x,y,z)   = -1;
            elseif  o1(x,y,z,6) == 1
                    c1(x,y,z)   = 3;
            elseif  o1(x,y,z,5) == 1
                    c1(x,y,z)   = 2;
            elseif  o1(x,y,z,4) == 1
                    c1(x,y,z)   = 1;
            end
        end
    end
    subplot(4,3,z);
    imshow(c1(:,:,z));
    colormap jet;
    caxis([-3,3]);

end



for z = 1:3
    for x = 1: size(w2,1)
        for y = 1: size(w2,2)
            if      o2(x,y,z,1) == 1
                    c2(x,y,z)   = -3;
            elseif  o2(x,y,z,2) == 1
                    c2(x,y,z)   = -2;
            elseif  o2(x,y,z,3) == 1
                    c2(x,y,z)   = -1;
```

```
        elseif  o2(x,y,z,6) == 1
                c2(x,y,z)   = 3;
        elseif  o2(x,y,z,5) == 1
                c2(x,y,z)   = 2;
        elseif  o2(x,y,z,4) == 1
                c2(x,y,z)   = 1;
        end
      end
   end
   subplot(4,3,z+3);
   imshow(c2(:,:,z));
   colormap jet;
   caxis([-3,3]);
end


for z = 1:3
   for x = 1: size(w3,1)
      for y = 1: size(w3,2)
        if      o3(x,y,z,1) == 1
                c3(x,y,z)   = -3;
        elseif  o3(x,y,z,2) == 1
                c3(x,y,z)   = -2;
        elseif  o3(x,y,z,3) == 1
                c3(x,y,z)   = -1;
        elseif  o3(x,y,z,6) == 1
                c3(x,y,z)   = 3;
        elseif  o3(x,y,z,5) == 1
                c3(x,y,z)   = 2;
        elseif  o3(x,y,z,4) == 1
                c3(x,y,z)   = 1;
        end
      end
   end
   subplot(4,3,z+6);
   imshow(c3(:,:,z));
   colormap jet;
   caxis([-3,3]);
end

for z = 1:3
   for x = 1: size(w4,1)
      for y = 1: size(w4,2)
        if      o4(x,y,z,1) == 1
                c4(x,y,z)   = -3;
        elseif  o4(x,y,z,2) == 1
                c4(x,y,z)   = -2;
        elseif  o4(x,y,z,3) == 1
                c4(x,y,z)   = -1;
        elseif  o4(x,y,z,6) == 1
                c4(x,y,z)   = 3;
        elseif  o4(x,y,z,5) == 1
                c4(x,y,z)   = 2;
        elseif  o4(x,y,z,4) == 1
                c4(x,y,z)   = 1;
```

```
        end
      end
    end
    subplot(4,3,z+9);
    imshow(c4(:,:,z));
    colormap jet;
    caxis([-3,3]);
end

%Save data as Structure

w1=struct('CmPixels',dist, 'Cropped', w1, 'RedMean', rmean1, 'RedSTD', rstd1, 'GreenMean', gmean1,
'GreenSTD', gstd1, ...
    'BlueMean', bmean1, 'BlueSTD', bstd1, 'OutputImage', c1, 'OutputArea', n1(:,:,1) );

w2=struct('CmPixels',dist, 'Cropped', w2, 'RedMean', rmean2, 'RedSTD', rstd2, 'GreenMean', gmean2,
'GreenSTD', gstd2, ...
    'BlueMean', bmean2, 'BlueSTD', bstd2, 'OutputImage', c2, 'OutputArea', n1(:,:,2) );

w3=struct('CmPixels',dist, 'Cropped', w3, 'RedMean', rmean3, 'RedSTD', rstd3, 'GreenMean', gmean3,
'GreenSTD', gstd3, ...
    'BlueMean', bmean3, 'BlueSTD', bstd3, 'OutputImage', c3, 'OutputArea', n1(:,:,3) );

w4=struct('CmPixels',dist, 'Cropped', w4, 'RedMean', rmean4, 'RedSTD', rstd4, 'GreenMean', gmean4,
'GreenSTD', gstd4, ...
    'BlueMean', bmean4, 'BlueSTD', bstd4, 'OutputImage', c4, 'OutputArea', n1(:,:,4) );

output=struct('Wound1', w1,'Wound2',w2,'Wound3',w3,'Wound4', w4);

%Save Figure
file = sprintf(['/Users/Peter/Desktop/pig/kinect/data/png/' name '.png']);
saveas(gcf,file);


%Save Surface Areas as .xls

xlswrite(['C:\Users\Peter\Desktop\pig\kinect\data\xls\' name '.xlsx'],n1(:,:,1));
xlswrite(['C:\Users\Peter\Desktop\pig\kinect\data\xls\' name '.xlsx'],n1(:,:,2),1,'A5');
xlswrite(['C:\Users\Peter\Desktop\pig\kinect\data\xls\' name '.xlsx'],n1(:,:,3),1,'A9');
xlswrite(['C:\Users\Peter\Desktop\pig\kinect\data\xls\' name '.xlsx'],n1(:,:,4),1,'A13');

eval([name '= output']);
save(sprintf(['/Users/Peter/Desktop/pig/kinect/data/batch/' name '.mat']),name);
end


function kinect_sa_batch()
%Run Batch Statistical Analysis on Kinect Images
running = 1;
%File Location
p = 'C:\Data\';
while running
    %Load Data
```

```
    [f,p]=uigetfile(strcat(p,'*.png'));
    image = imread([p f]);
    %Run SA Method
    skincalcmulti(image);
    button = questdlg('Keep Going?',' ','Yes','No','Yes');
    if strcmp('Yes',button);
        running = 1;
        close gcf;
    else
        running = 0;
    end
end
end
```

```
function output = kinectir_sa(image)
%Perform Statistical Analysis on Kinect NIR Images
%Select Four Wounds from Image and Four Skin Samples

disp('skin1');
skin1 = imcrop(image);
disp('w1');
w1   = imcrop(image);

disp('skin2');
skin2 = imcrop(image);
disp('w2');
w2= imcrop(image);

disp('skin3');
skin3 = imcrop(image);
disp('w3');
w3   = imcrop(image);

disp('skin4');
skin4 = imcrop(image);
disp('w4');
w4   = imcrop(image);

%Measure Lenght of 30 cm (Lenght of Ruler) and convert to pixels per cm

distline = imdistline;
pause;
meter = getDistance(distline);
filename = (inputdlg('Whats the name?'));
name = filename{1};
dist = meter/30;
close;

%Calculate Skin Means and STD
rskin1 = skin1(:,:,1);
```

```
gskin1 = skin1(:,:,2);
bskin1 = skin1(:,:,3);
rmean1 = mean2(rskin1);
gmean1 = mean2(gskin1);
bmean1 = mean2(bskin1);
rstd1  = std2(rskin1);
gstd1  = std2(gskin1);
bstd1  = std2(bskin1);

rskin2 = skin2(:,:,1);
gskin2 = skin2(:,:,2);
bskin2 = skin2(:,:,3);
rmean2 = mean2(rskin2);
gmean2 = mean2(gskin2);
bmean2 = mean2(bskin2);
rstd2  = std2(rskin2);
gstd2  = std2(gskin2);
bstd2  = std2(bskin2);

rskin3 = skin3(:,:,1);
gskin3 = skin3(:,:,2);
bskin3 = skin3(:,:,3);
rmean3 = mean2(rskin3);
gmean3 = mean2(gskin3);
bmean3 = mean2(bskin3);
rstd3  = std2(rskin3);
gstd3  = std2(gskin3);
bstd3  = std2(bskin3);

rskin4 = skin4(:,:,1);
gskin4 = skin4(:,:,2);
bskin4 = skin4(:,:,3);
rmean4 = mean2(rskin4);
gmean4 = mean2(gskin4);
bmean4 = mean2(bskin4);
rstd4  = std2(rskin4);
gstd4  = std2(gskin4);
bstd4  = std2(bskin4);



%Create Output Arrays

o1 = zeros(size(w1,1),size(w1,2),3,6);
o2 = zeros(size(w2,1),size(w2,2),3,6);
o3 = zeros(size(w3,1),size(w3,2),3,6);
o4 = zeros(size(w4,1),size(w4,2),3,6);

%Cycle Through Pixels in image, compare to postivive STD Tresholds then negative,
%.. if True set Output Array value to 1 (or True)
for x = 1:size(w1,1)
   for y = 1:size(w1,2)
      for z = 1:3
         if w1(x,y,1) > (rmean1 + z*rstd1)
```

```
            o1(x,y,1,z+3) = 1;
        end
        if w1(x,y,2) > (gmean1 + z*gstd1)
            o1(x,y,2,z+3) = 1;
        end
        if w1(x,y,3) > (bmean1 + z*bstd1)
            o1(x,y,3,z+3) = 1;
        end
    end
    for z = -3:-1
        if w1(x,y,1) < (rmean1 + z*rstd1)
            o1(x,y,1,z+4) = 1;
        end
        if w1(x,y,2) < (gmean1 + z*gstd1)
            o1(x,y,2,z+4) = 1;
        end
        if w1(x,y,3) < (bmean1 + z*bstd1)
            o1(x,y,3,z+4) = 1;
        end
    end
  end
end


for x = 1:size(w2,1)
  for y = 1:size(w2,2)
    for z = 1:3
        if w2(x,y,1) > (rmean2 + z*rstd2)
            o2(x,y,1,z+3) = 1;
        end
        if w2(x,y,2) > (gmean2 + z*gstd2)
            o2(x,y,2,z+3) = 1;
        end
        if w2(x,y,3) > (bmean2 + z*bstd2)
            o2(x,y,3,z+3) = 1;
        end
    end
    for z = -3:-1
        if w2(x,y,1) < (rmean2 + z*rstd2)
            o2(x,y,1,z+4) = 1;
        end
        if w2(x,y,2) < (gmean2 + z*gstd2)
            o2(x,y,2,z+4) = 1;
        end
        if w2(x,y,3) < (bmean2 + z*bstd2)
            o2(x,y,3,z+4) = 1;
        end
    end
  end
end

for x = 1:size(w3,1)
  for y = 1:size(w3,2)
    for z = 1:3
        if w3(x,y,1) > (rmean3 + z*rstd3)
```

```
            o3(x,y,1,z+3) = 1;
        end
        if w3(x,y,2) > (gmean3 + z*gstd3)
            o3(x,y,2,z+3) = 1;
        end
        if w3(x,y,3) > (bmean3 + z*bstd3)
            o3(x,y,3,z+3) = 1;
        end
    end
    for z = -3:-1
        if w3(x,y,1) < (rmean3 + z*rstd3)
            o3(x,y,1,z+4) = 1;
        end
        if w3(x,y,2) < (gmean3 + z*gstd3)
            o3(x,y,2,z+4) = 1;
        end
        if w3(x,y,3) < (bmean3 + z*bstd3)
            o3(x,y,3,z+4) = 1;
        end
    end
  end
end

for x = 1:size(w4,1)
  for y = 1:size(w4,2)
    for z = 1:3
        if w4(x,y,1) > (rmean4 + z*rstd4)
            o4(x,y,1,z+3) = 1;
        end
        if w4(x,y,2) > (gmean4 + z*gstd4)
            o4(x,y,2,z+3) = 1;
        end
        if w4(x,y,3) > (bmean4 + z*bstd4)
            o4(x,y,3,z+3) = 1;
        end
    end
    for z = -3:-1
        if w4(x,y,1) < (rmean4 + z*rstd4)
            o4(x,y,1,z+4) = 1;
        end
        if w4(x,y,2) < (gmean4 + z*gstd4)
            o4(x,y,2,z+4) = 1;
        end
        if w4(x,y,3) < (bmean4 + z*bstd4)
            o4(x,y,3,z+4) = 1;
        end
    end
  end
end

%Create Array and Calcuate Surface Areas for Each STD Threshold
  n1 = zeros(3,6,4);
  for d = 1:6
    for c = 1:3
        n1(c,d,1) = (sum(sum(o1(:,:,c,d)))/(dist^2));
```

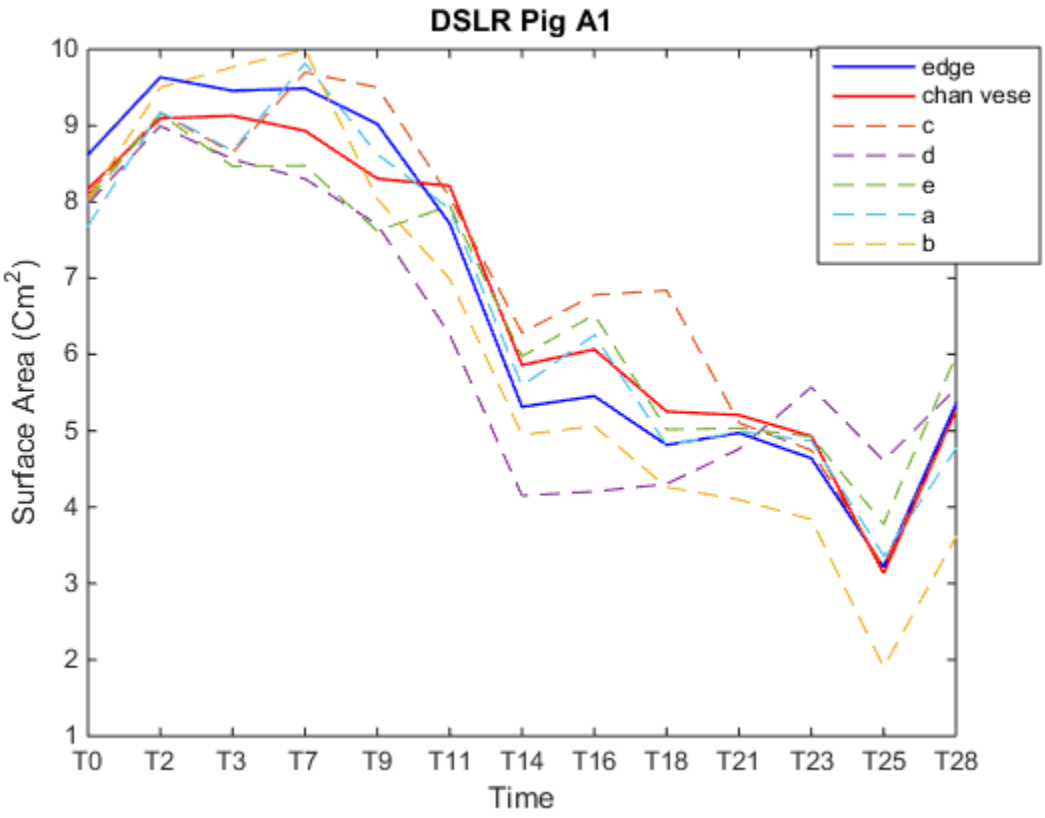
```
    end
  end

  for d = 1:6
    for c = 1:3
      n1(c,d,2) = (sum(sum(o2(:,:,c,d)))/(dist^2));
    end
  end


  for d = 1:6
    for c = 1:3
      n1(c,d,3) = (sum(sum(o3(:,:,c,d)))/(dist^2));
    end
  end


  for d = 1:6
    for c = 1:3
      n1(c,d,4) = (sum(sum(o4(:,:,c,d)))/(dist^2));
    end
  end


  %Create Combined Output Display
c1 = zeros(size(w1,1),size(w1,2),3);
c2 = zeros(size(w2,1),size(w2,2),3);
c3 = zeros(size(w3,1),size(w3,2),3);
c4 = zeros(size(w4,1),size(w4,2),3);

%Combine the Six Output Masks into one mask per color channel

for z = 1:3
  for x = 1: size(w1,1)
    for y = 1: size(w1,2)
      if    o1(x,y,z,1) == 1
            c1(x,y,z)   = -3;
      elseif  o1(x,y,z,2) == 1
            c1(x,y,z)   = -2;
      elseif  o1(x,y,z,3) == 1
            c1(x,y,z)   = -1;
      elseif  o1(x,y,z,6) == 1
            c1(x,y,z)   = 3;
      elseif  o1(x,y,z,5) == 1
            c1(x,y,z)   = 2;
      elseif  o1(x,y,z,4) == 1
            c1(x,y,z)   = 1;
      end
    end
  end
  subplot(4,3,z);
  imshow(c1(:,:,z));
  colormap jet;
  caxis([-3,3]);
```

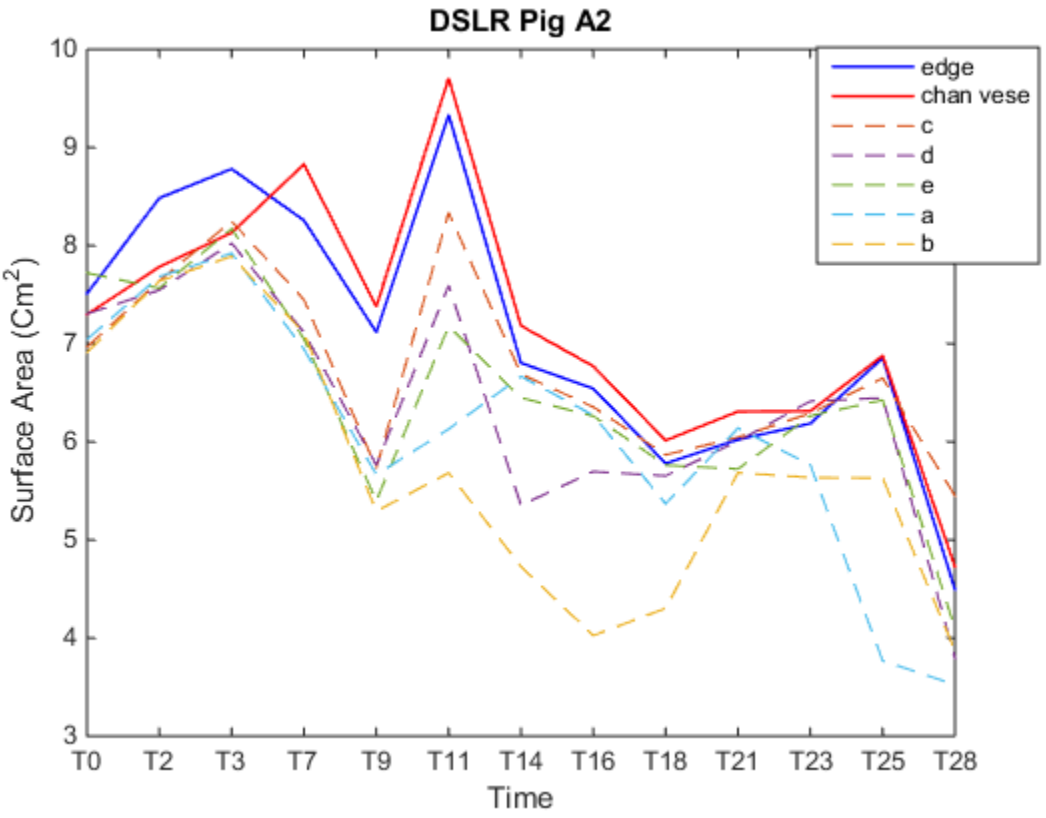
```
end



for z = 1:3
   for x = 1: size(w2,1)
      for y = 1: size(w2,2)
         if      o2(x,y,z,1) == 1
                 c2(x,y,z)   = -3;
         elseif  o2(x,y,z,2) == 1
                 c2(x,y,z)   = -2;
         elseif  o2(x,y,z,3) == 1
                 c2(x,y,z)   = -1;
         elseif  o2(x,y,z,6) == 1
                 c2(x,y,z)   = 3;
         elseif  o2(x,y,z,5) == 1
                 c2(x,y,z)   = 2;
         elseif  o2(x,y,z,4) == 1
                 c2(x,y,z)   = 1;
         end
      end
   end
   subplot(4,3,z+3);
   imshow(c2(:,:,z));
   colormap jet;
   caxis([-3,3]);
end



for z = 1:3
   for x = 1: size(w3,1)
      for y = 1: size(w3,2)
         if      o3(x,y,z,1) == 1
                 c3(x,y,z)   = -3;
         elseif  o3(x,y,z,2) == 1
                 c3(x,y,z)   = -2;
         elseif  o3(x,y,z,3) == 1
                 c3(x,y,z)   = -1;
         elseif  o3(x,y,z,6) == 1
                 c3(x,y,z)   = 3;
         elseif  o3(x,y,z,5) == 1
                 c3(x,y,z)   = 2;
         elseif  o3(x,y,z,4) == 1
                 c3(x,y,z)   = 1;
         end
      end
   end
   subplot(4,3,z+6);
   imshow(c3(:,:,z));
   colormap jet;
   caxis([-3,3]);
end
```

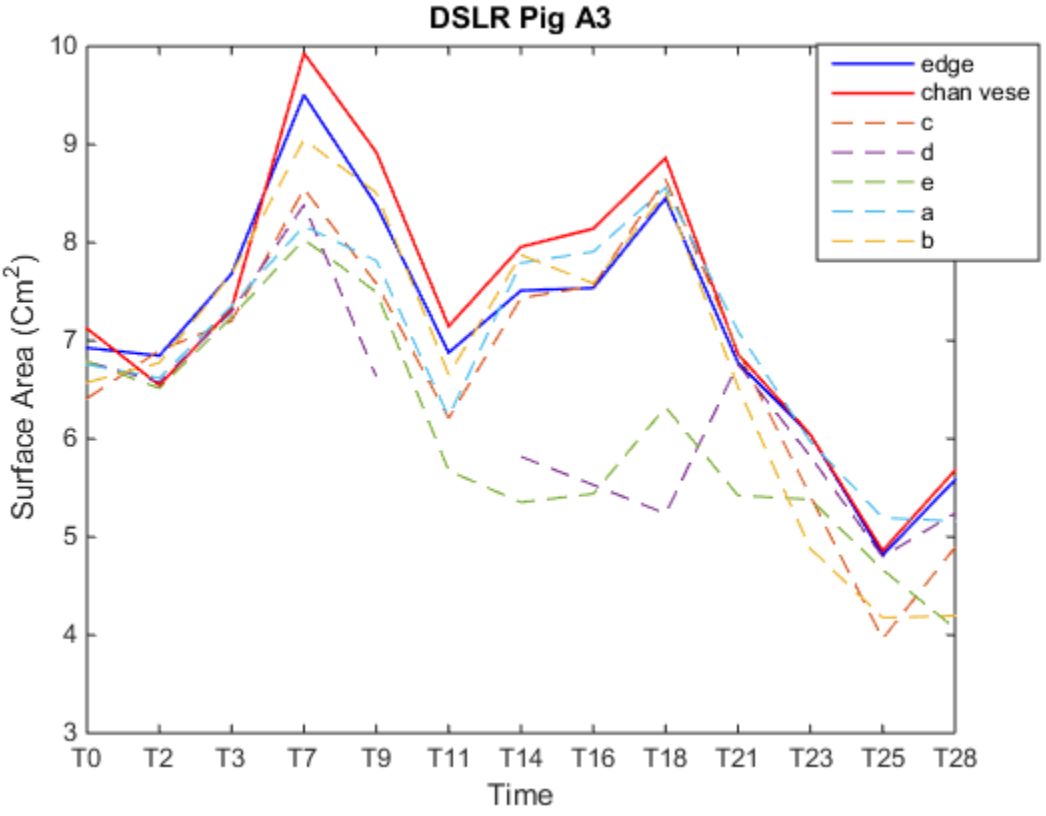
```
for z = 1:3
   for x = 1: size(w4,1)
      for y = 1: size(w4,2)
         if      o4(x,y,z,1) == 1
                 c4(x,y,z)   = -3;
         elseif  o4(x,y,z,2) == 1
                 c4(x,y,z)   = -2;
         elseif  o4(x,y,z,3) == 1
                 c4(x,y,z)   = -1;
         elseif  o4(x,y,z,6) == 1
                 c4(x,y,z)   = 3;
         elseif  o4(x,y,z,5) == 1
                 c4(x,y,z)   = 2;
         elseif  o4(x,y,z,4) == 1
                 c4(x,y,z)   = 1;
         end
      end
   end
   subplot(4,3,z+9);
   imshow(c4(:,:,z));
   colormap jet;
   caxis([-3,3]);
end

%Save data as Structure

w1=struct('CmPixels',dist, 'Cropped', w1, 'RedMean', rmean1, 'RedSTD', rstd1, 'GreenMean', gmean1,
'GreenSTD', gstd1, ...
   'BlueMean', bmean1, 'BlueSTD', bstd1, 'OutputImage', gcf, 'OutputArea', n1(:,:,1) );

w2=struct('CmPixels',dist, 'Cropped', w2, 'RedMean', rmean2, 'RedSTD', rstd2, 'GreenMean', gmean2,
'GreenSTD', gstd2, ...
   'BlueMean', bmean2, 'BlueSTD', bstd2, 'OutputImage', gcf, 'OutputArea', n1(:,:,2) );

w3=struct('CmPixels',dist, 'Cropped', w3, 'RedMean', rmean3, 'RedSTD', rstd3, 'GreenMean', gmean3,
'GreenSTD', gstd3, ...
   'BlueMean', bmean3, 'BlueSTD', bstd3, 'OutputImage', gcf, 'OutputArea', n1(:,:,3) );

w4=struct('CmPixels',dist, 'Cropped', w4, 'RedMean', rmean4, 'RedSTD', rstd4, 'GreenMean', gmean4,
'GreenSTD', gstd4, ...
   'BlueMean', bmean4, 'BlueSTD', bstd4, 'OutputImage', gcf, 'OutputArea', n1(:,:,4) );

output=struct('Wound1', w1,'Wound2',w2,'Wound3',w3,'Wound4', w4);
%Save Figure
file = sprintf(['/Users/Peter/Desktop/pig/kinect/data/png/' name '.png']);
saveas(gcf,file);

   %Save Surface Areas as .xls


xlswrite(['C:\Users\Peter\Desktop\pig\kinect\data\xls\' name '.xlsx'],n1(:,:,1));
xlswrite(['C:\Users\Peter\Desktop\pig\kinect\data\xls\' name '.xlsx'],n1(:,:,2),1,'A5');
xlswrite(['C:\Users\Peter\Desktop\pig\kinect\data\xls\' name '.xlsx'],n1(:,:,3),1,'A9');
xlswrite(['C:\Users\Peter\Desktop\pig\kinect\data\xls\' name '.xlsx'],n1(:,:,4),1,'A13');
```

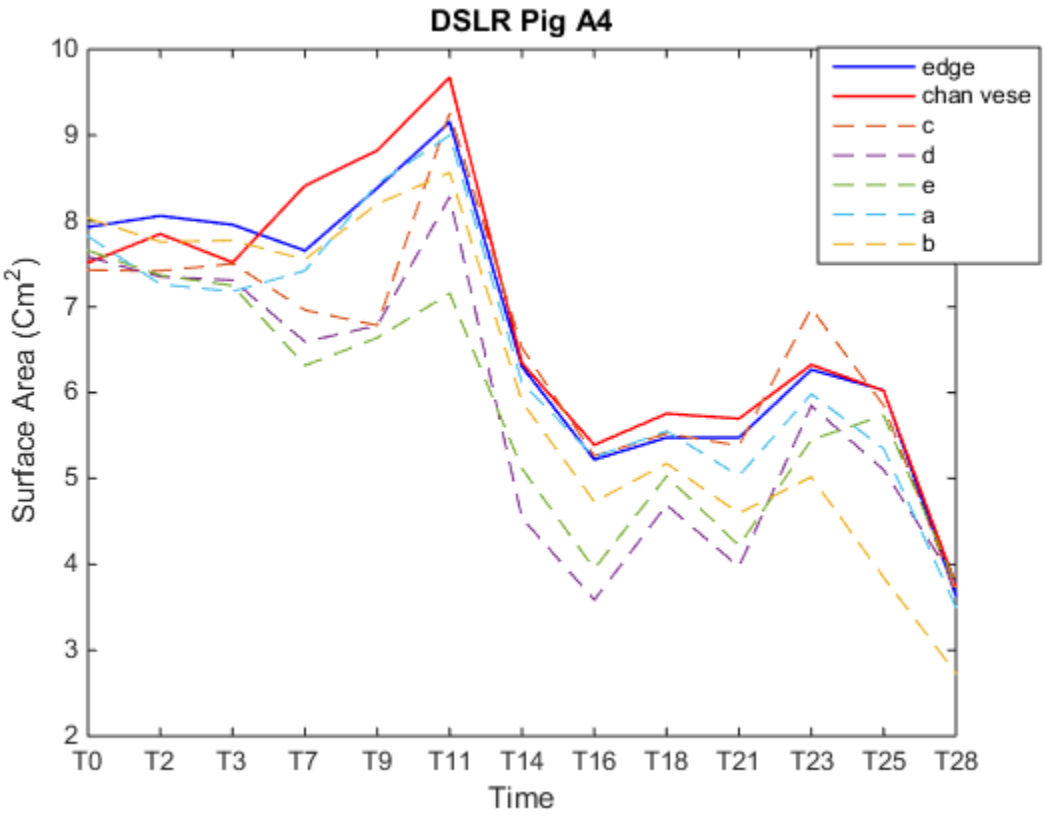
```
eval([name '= output']);
save(sprintf(['/Users/Peter/Desktop/pig/kinect/data/batch/' name '.mat']),name);
end

function output = kinectir_sa_batch()
%Run Batch Statistical Analysis on Kinect NIR Images
running = 1;
%Data Location
p = 'C:\Data\';
while running
   %Load Data
   [f,p]=uigetfile(strcat(p,'*.png'));
   image = imread([p f]);
   %Create 2 empty layers to make fake RGB images
   blank = zeros(424,512);
   blank = blank + 25000;
   image = cat(3,blank,image,blank);
   %Run Statistical Analysis
   output = skincalcmulti(image);
   button = questdlg('Keep Going?',' ','Yes','No','Yes');
   if strcmp('Yes',button);
      running = 1;
      close gcf;
   else
      running = 0;
   end
end
end

function m =  mask(image)
%Function to Create Initial Mask
hold on
imshow(image);
%Set Window Size to enlarge smaller wounds
set(gcf,'position',[200 100 850 750])
%Create a inital Mask
[sx, sy] = getpts;

n = size(sx,1);
n=n+1;
sxy=[];
sxy=[sx sy];
sxy=sxy';
sxy(:,n) = [sxy(1,1);sxy(2,1)];

% Interpolate Points and create Spline
t = 1:n;
ts = 1: 0.1: n;
xys = spline(t,sxy,ts);
%Convert Spline to Mask
bx = xys(1,:);
by = xys(2,:);
m=poly2mask(bx,by,size(image,1),size(image,2));
```

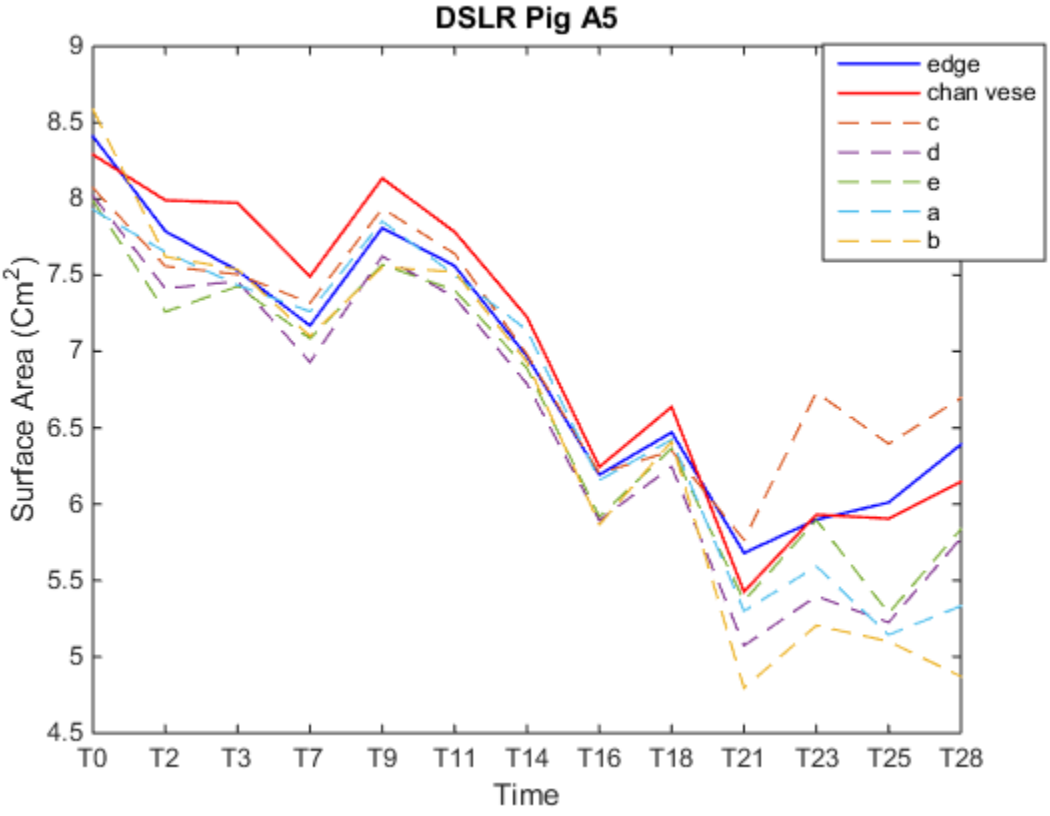
APPENDIX B: SURFACE AREA OUTPUTS FOR ACTIVE CONTOURS

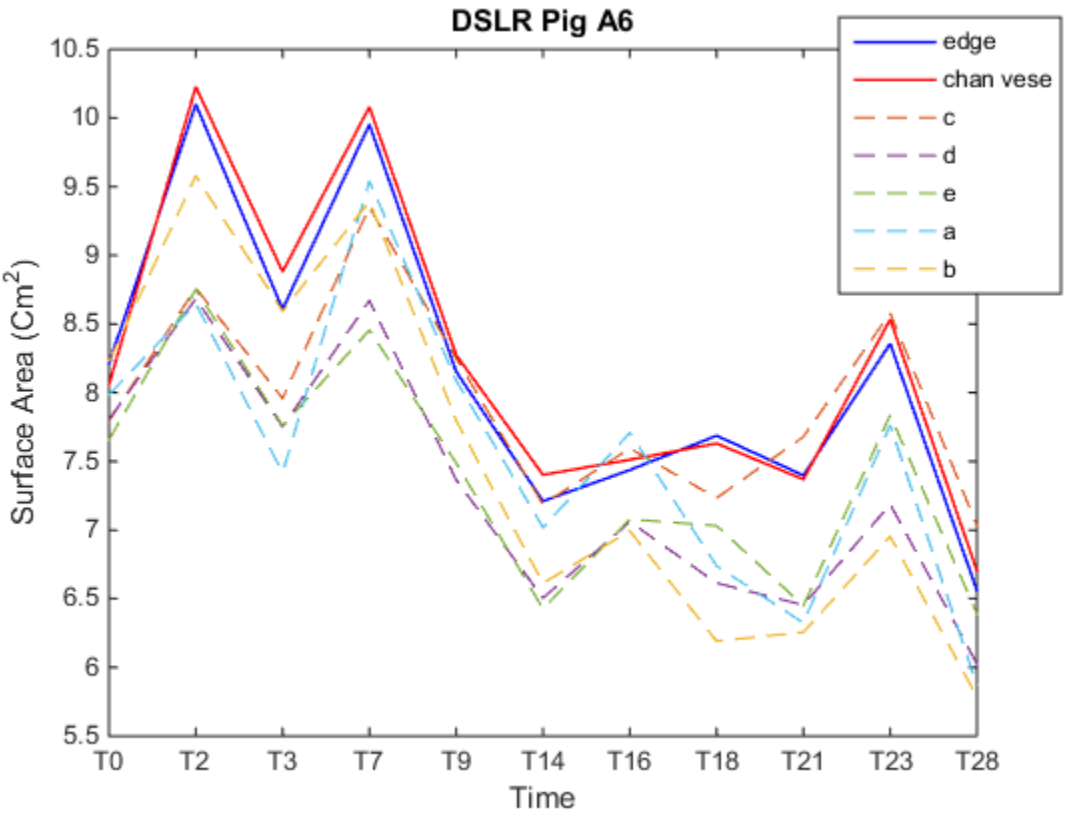


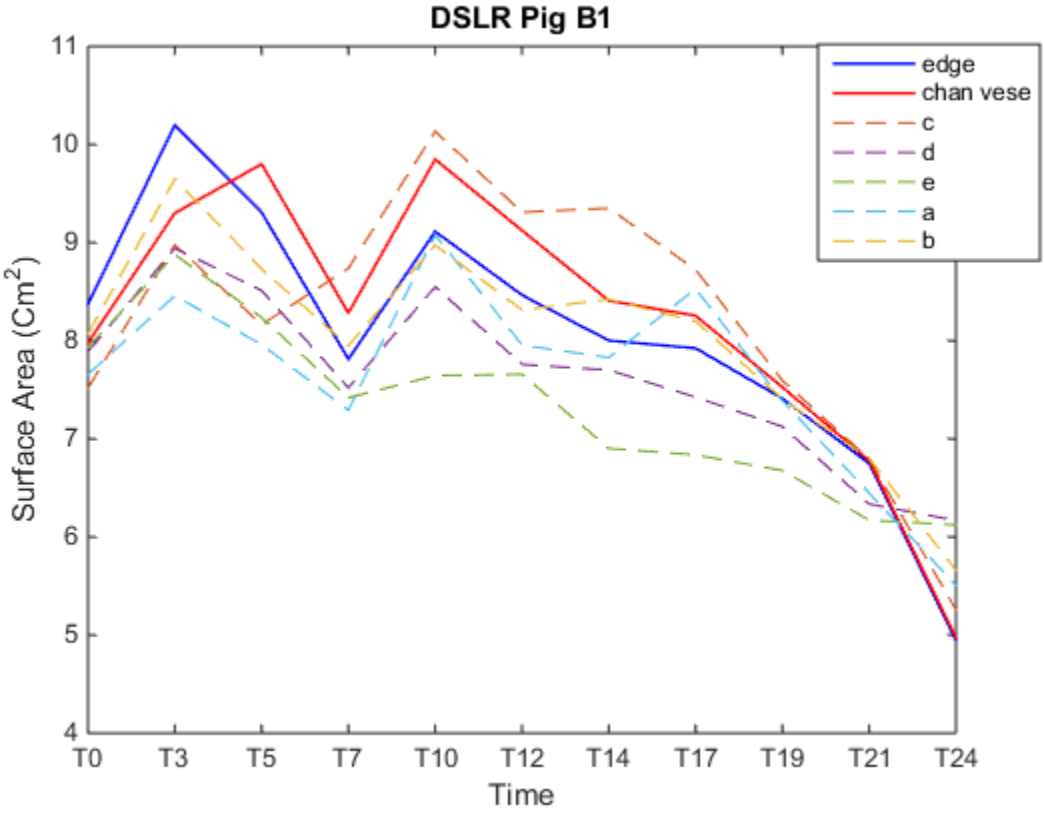

DSLR Pig A2
Surface Area (Cm2)
Time
edge
chan vese
c
d
e
a
b
10
9
8
7
6
5
4
3
T0
T2
T3
T7
T9
T11
T14
T16
T18
T21
T23
T25
T28

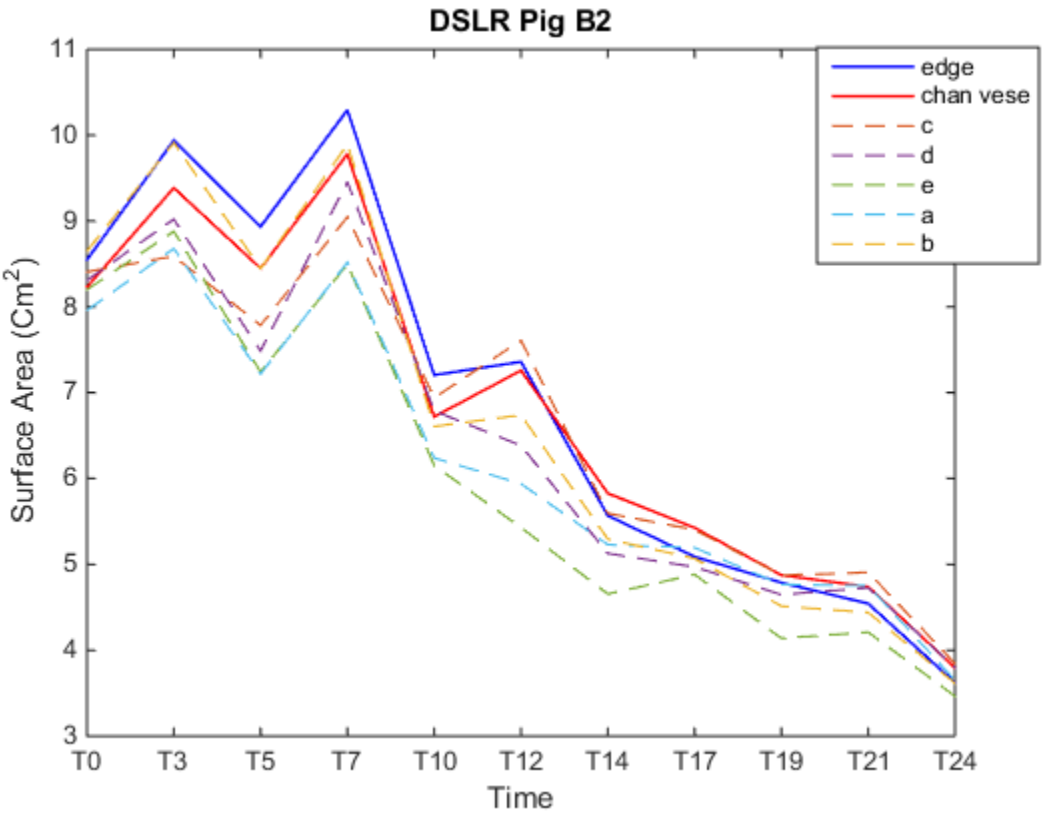

DSLR Pig A3
Surface Area (Cm2)
Time
edge
chan vese
c
d
e
a
b
10
9
8
7
6
5
4
3
T0
T2
T3
T7
T9
T11
T14
T16
T18
T21
T23
T25
T28

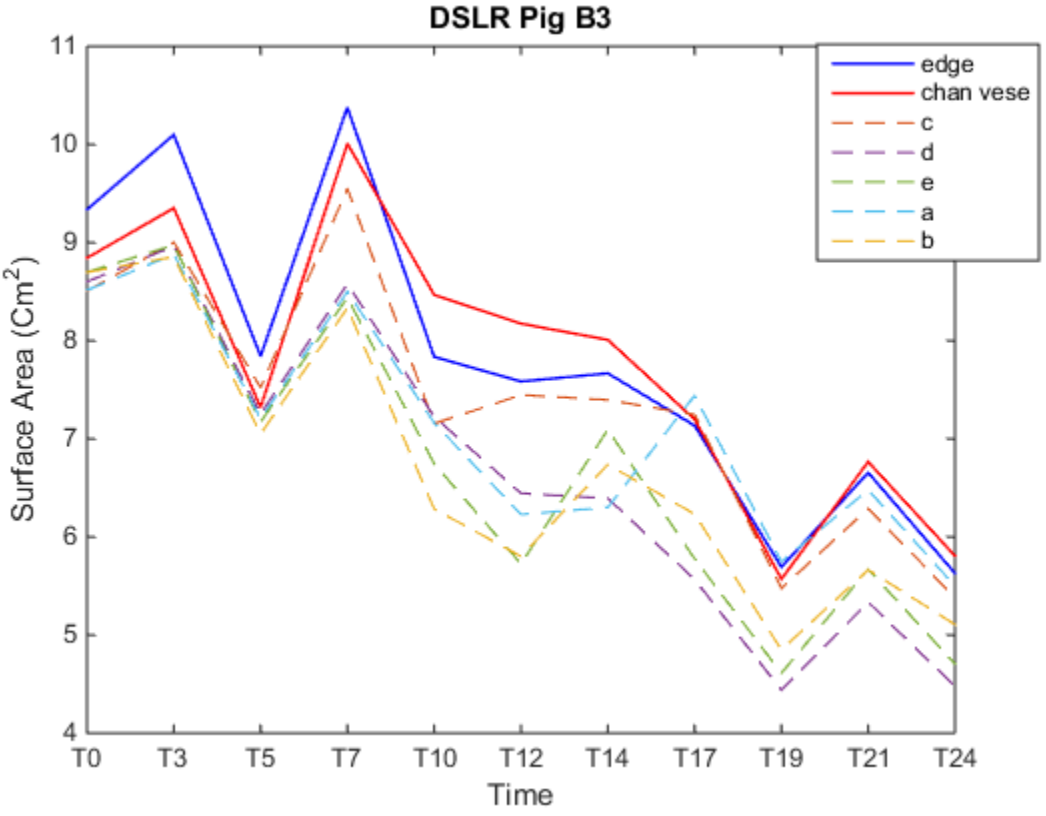

DSLR Pig A4
edge
chan vese
c
d
e
a
b
Surface Area (Cm2)
10
9
8
7
6
5
4
3
2
T0
T2
T3
T7
T9
T11
T14
T16
T18
T21
T23
T25
T28
Time

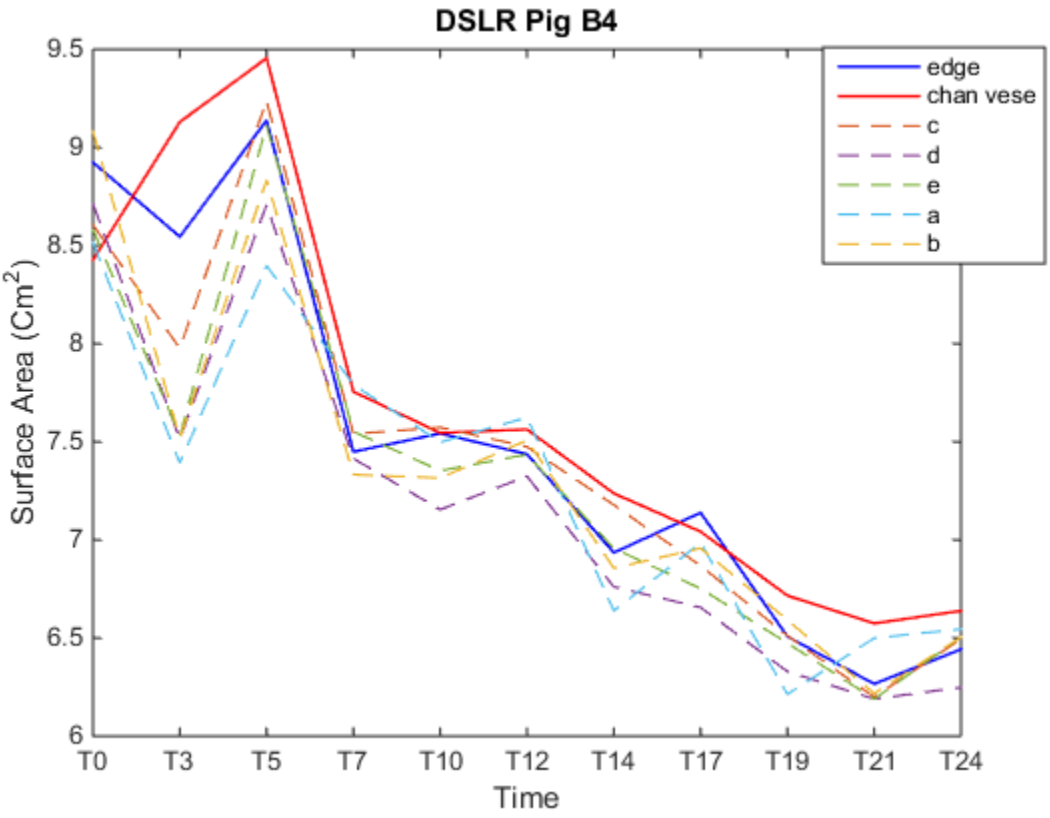

DSLR Pig A5
edge
chan vese
c
d
e
a
b
Surface Area (Cm2)
9
8.5
8
7.5
7
6.5
6
5.5
5
4.5
T0
T2
T3
T7
T9
T11
T14
T16
T18
T21
T23
T25
T28
Time

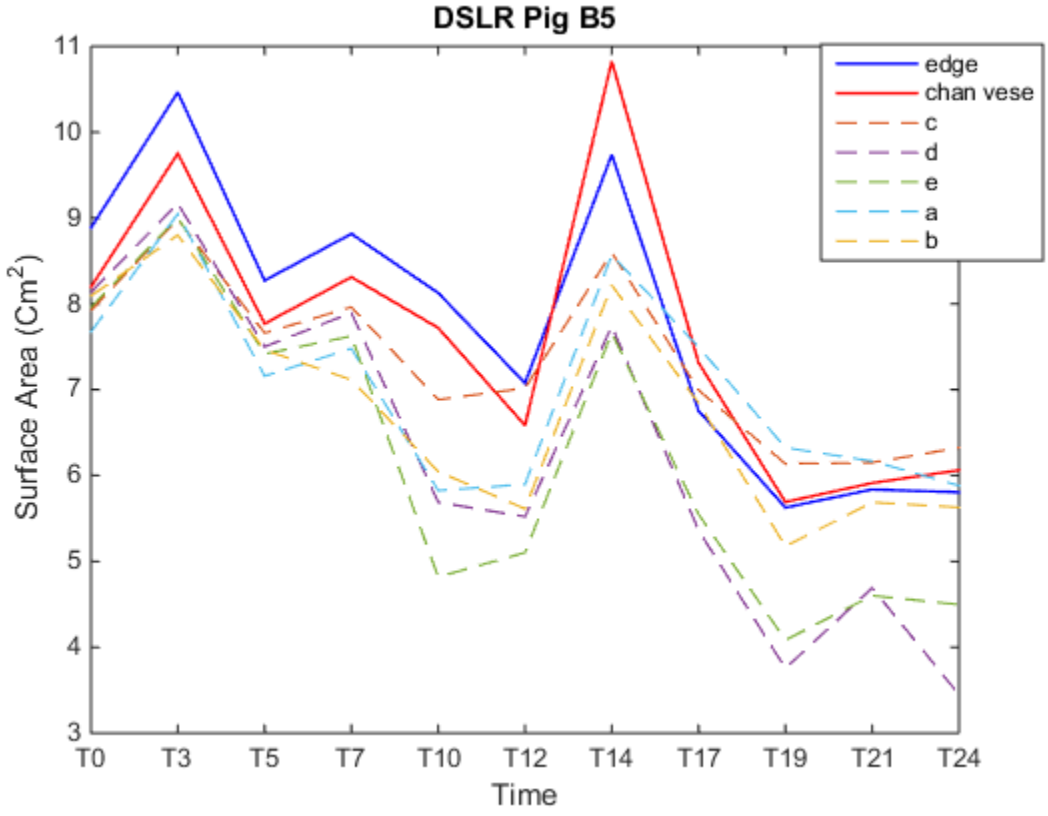

DSLR Pig A6
edge
chan vese
c
d
e
a
b
Surface Area (Cm2)
10.5
10
9.5
9
8.5
8
7.5
7
6.5
6
5.5
T0
T2
T3
T7
T9
T14
T16
T18
T21
T23
T28
Time

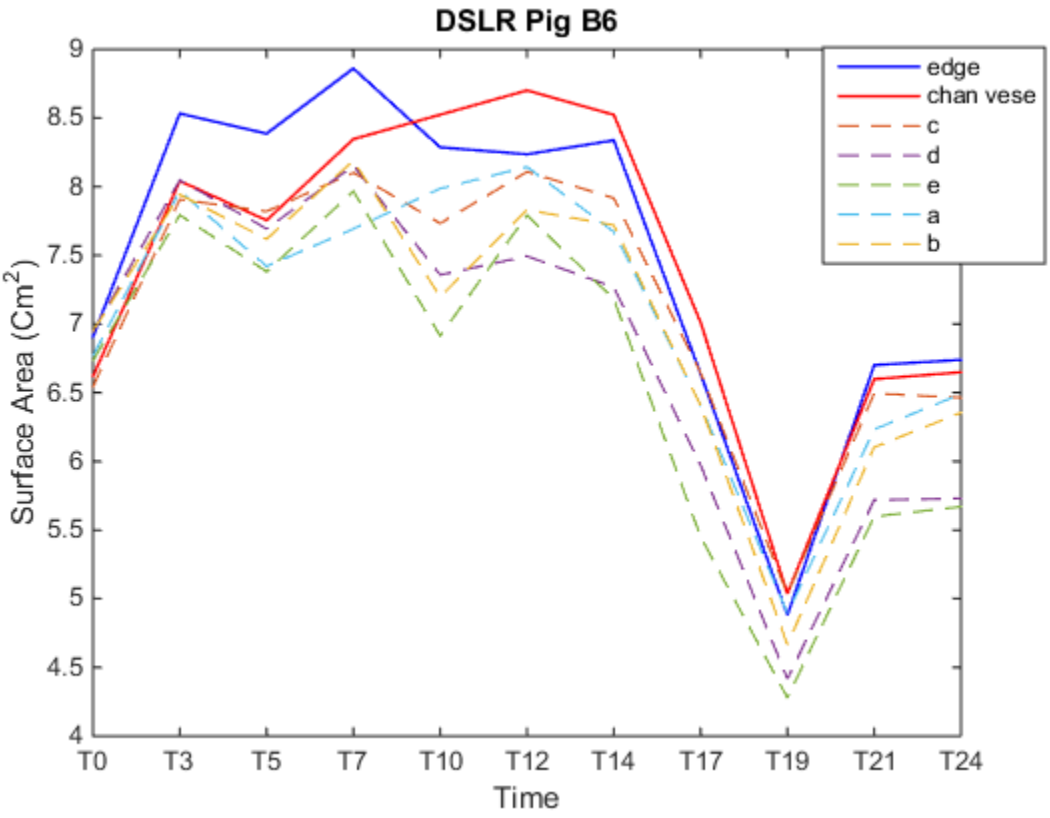

DSLR Pig B1
edge
chan vese
c
d
e
a
b
Surface Area (Cm2)
11
10
9
8
7
6
5
4
T0
T3
T5
T7
T10
T12
T14
T17
T19
T21
T24
Time

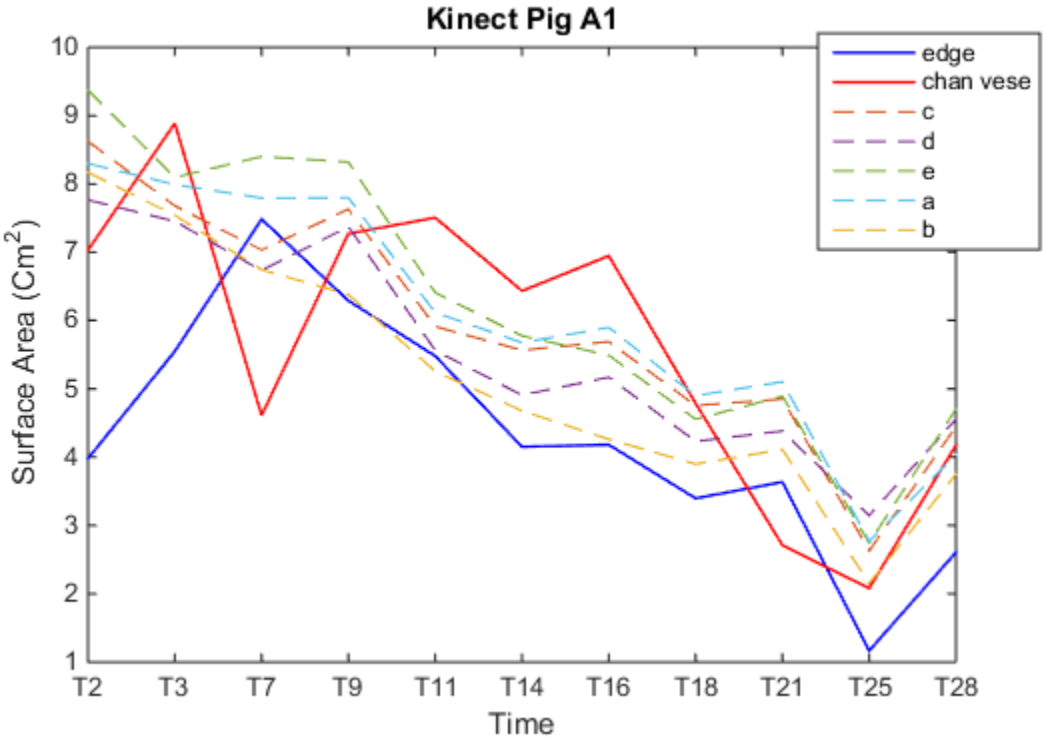

DSLR Pig B2
edge
chan vese
c
d
e
a
b
Surface Area (Cm^2)
Time
T0
T3
T5
T7
T10
T12
T14
T17
T19
T21
T24
3
4
5
6
7
8
9
10
11

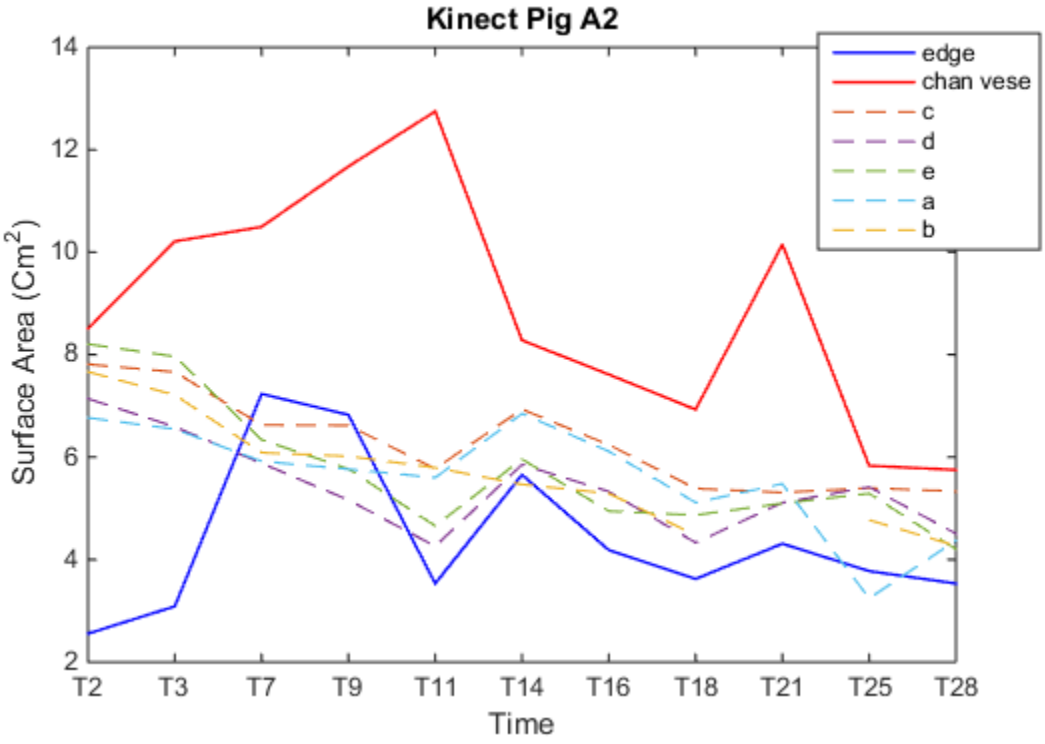

DSLR Pig B3
edge
chan vese
c
d
e
a
b
Surface Area (Cm^2)
Time
T0
T3
T5
T7
T10
T12
T14
T17
T19
T21
T24
4
5
6
7
8
9
10
11

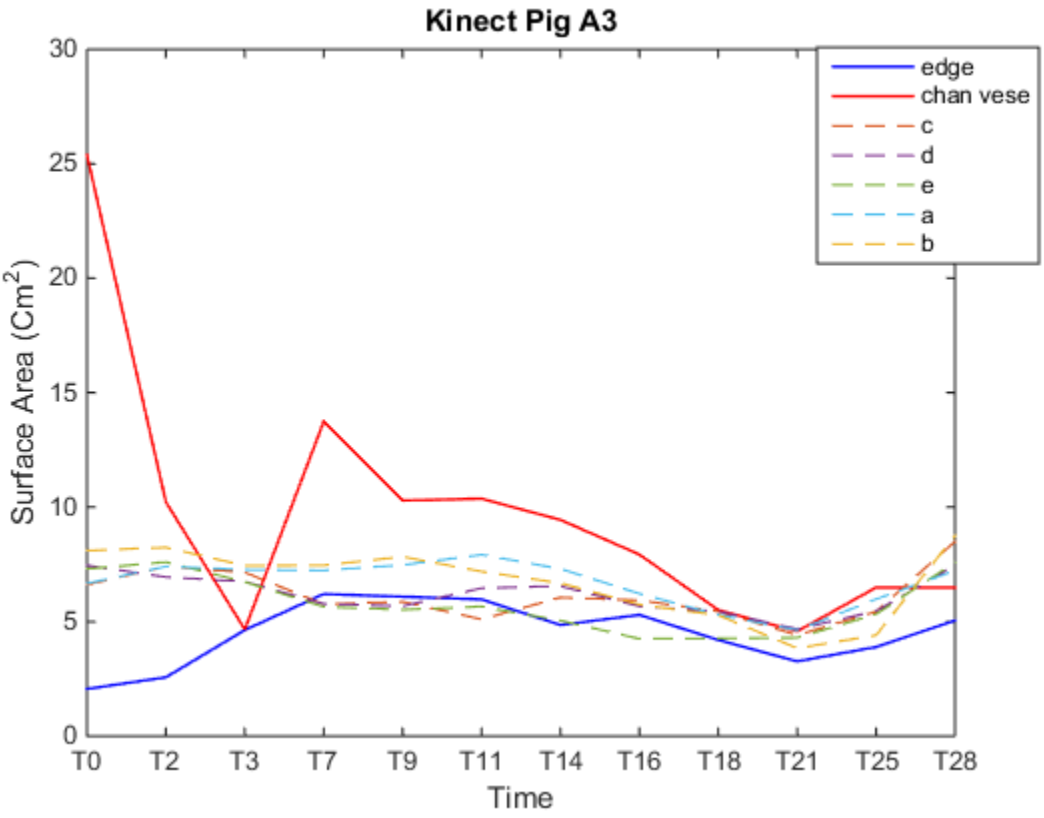

DSLR Pig B4
Surface Area (Cm^2)
Time
edge
chan vese
c
d
e
a
b
9.5
9
8.5
8
7.5
7
6.5
6
T0
T3
T5
T7
T10
T12
T14
T17
T19
T21
T24

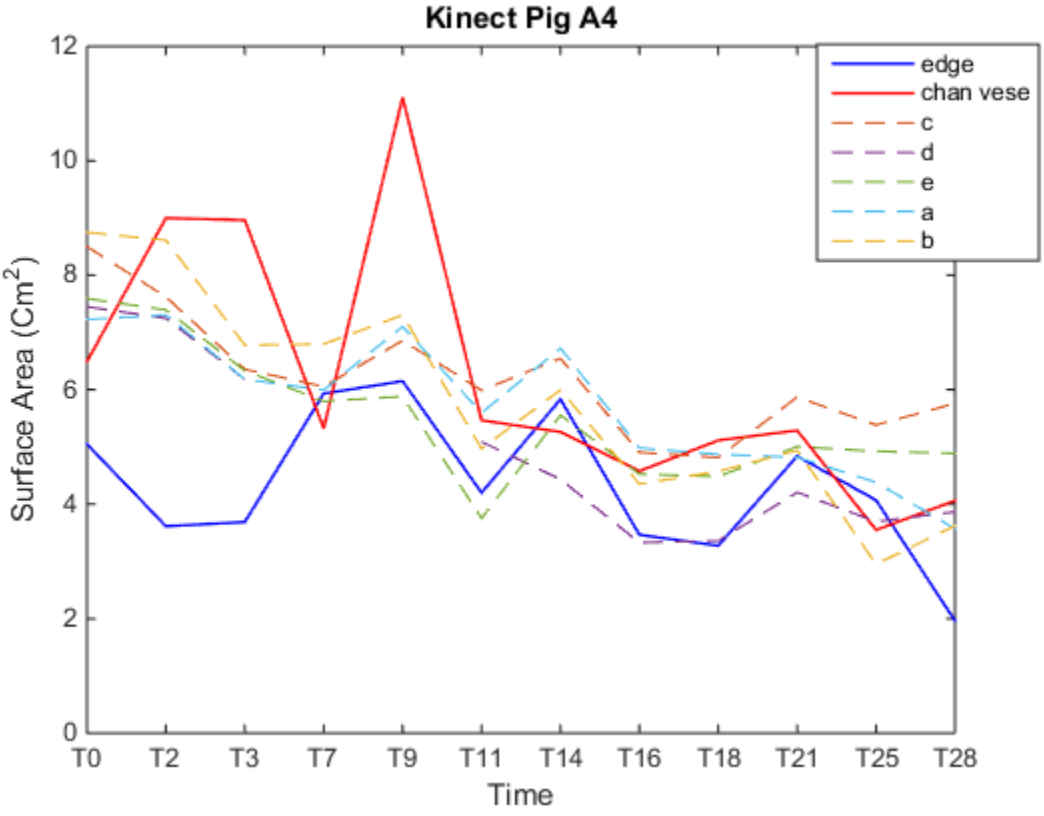

DSLR Pig B5
Surface Area (Cm^2)
Time
edge
chan vese
c
d
e
a
b
11
10
9
8
7
6
5
4
3
T0
T3
T5
T7
T10
T12
T14
T17
T19
T21
T24

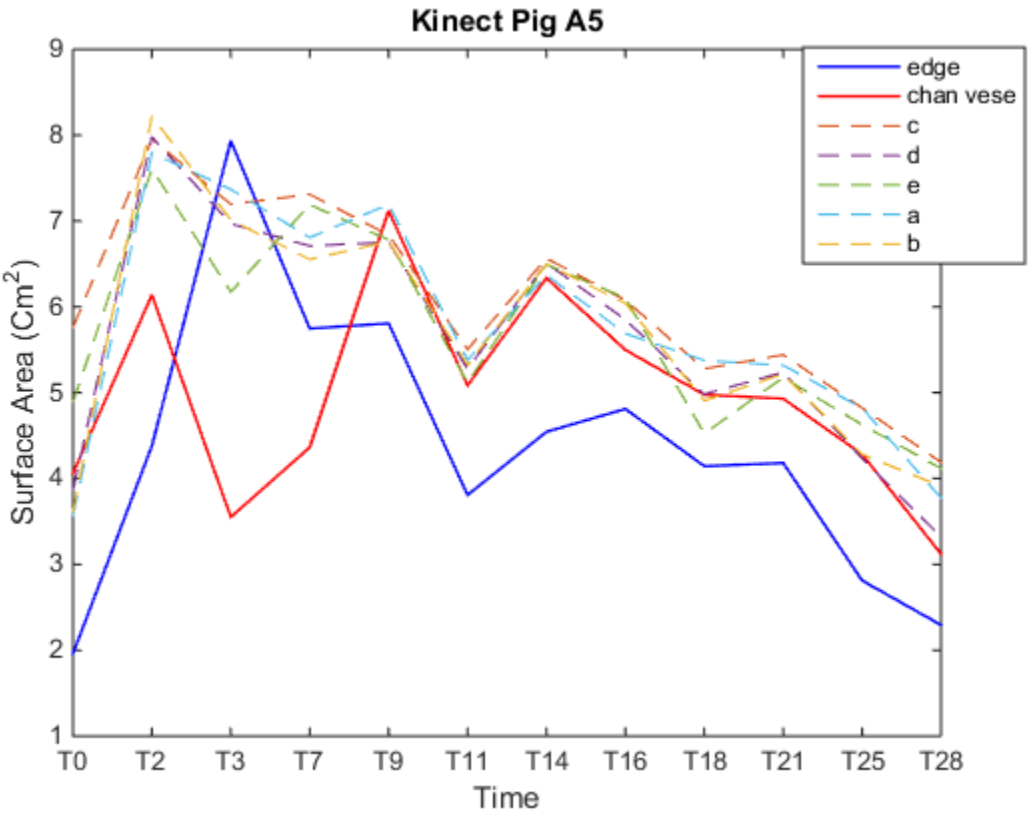

DSLR Pig B6
edge
chan vese
c
d
e
a
b
Surface Area (Cm2)
9
8.5
8
7.5
7
6.5
6
5.5
5
4.5
4
T0
T3
T5
T7
T10
T12
T14
T17
T19
T21
T24
Time

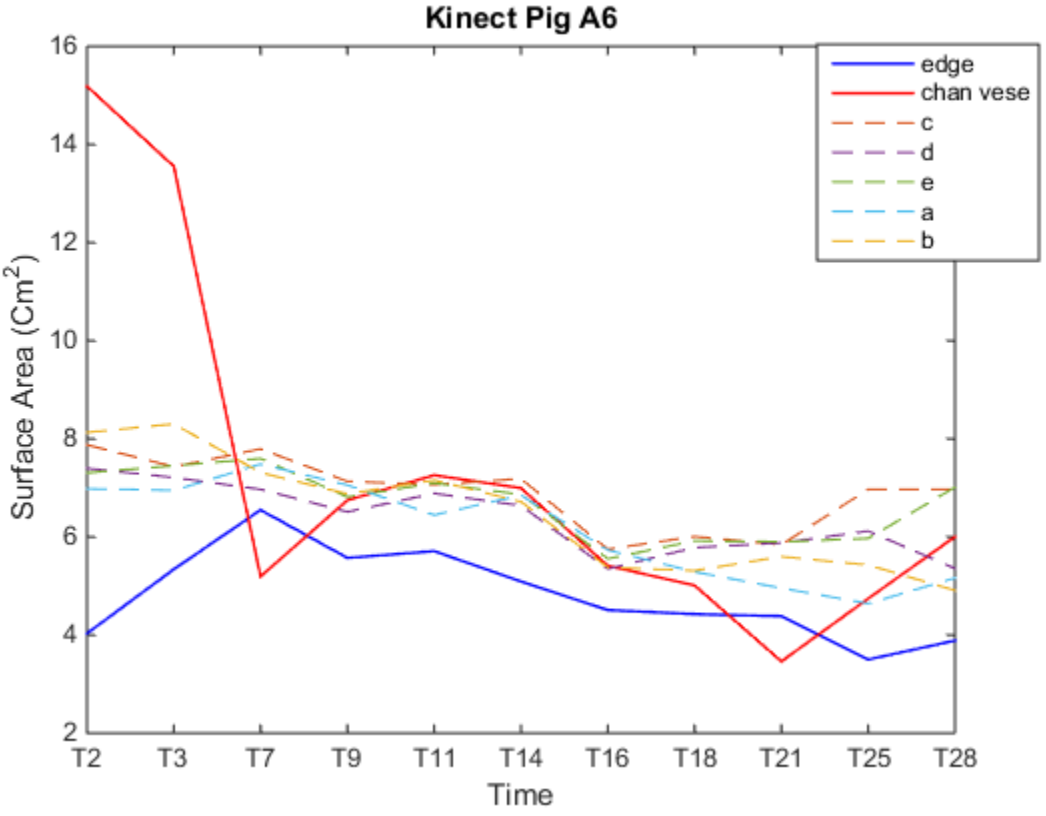

Kinect Pig A1
edge
chan vese
c
d
e
a
b
Surface Area (Cm^2)
Time
T2
T3
T7
T9
T11
T14
T16
T18
T21
T25
T28
1
2
3
4
5
6
7
8
9
10

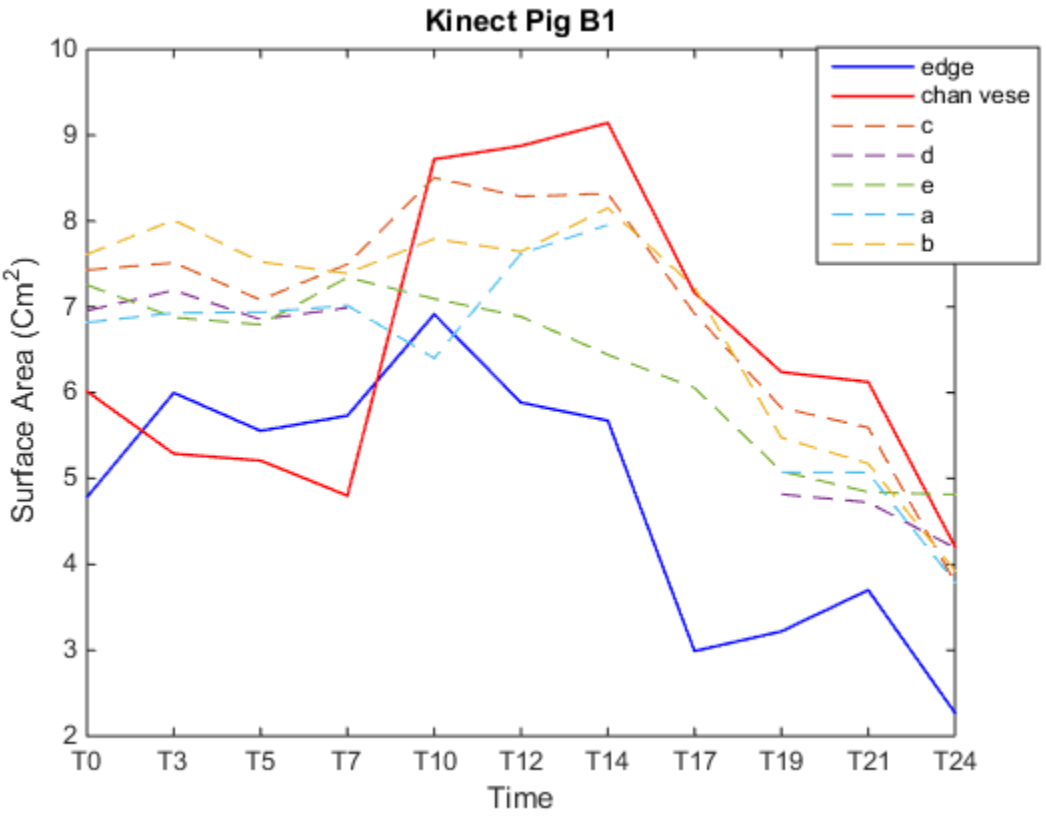

Kinect Pig A2
edge
chan vese
c
d
e
a
b
Surface Area (Cm^2)
Time
T2
T3
T7
T9
T11
T14
T16
T18
T21
T25
T28
2
4
6
8
10
12
14

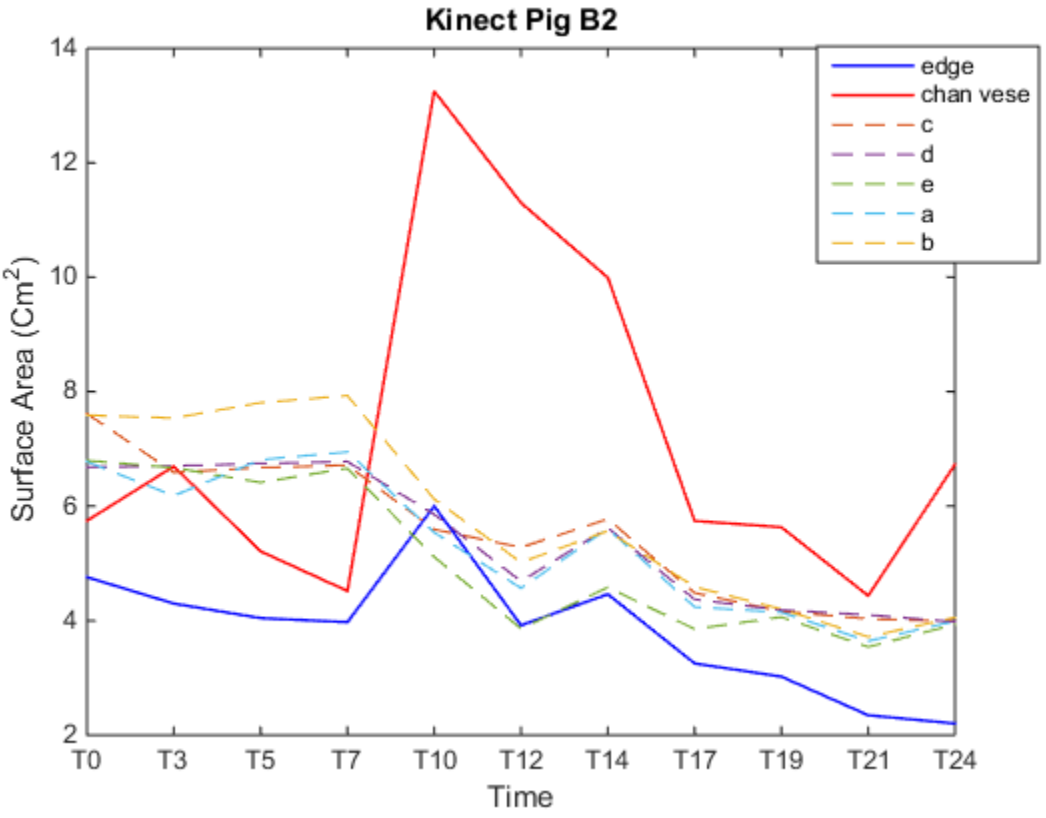

Kinect Pig A3
edge
chan vese
c
d
e
a
b
Surface Area (Cm2)
0
5
10
15
20
25
30
T0
T2
T3
T7
T9
T11
T14
T16
T18
T21
T25
T28
Time

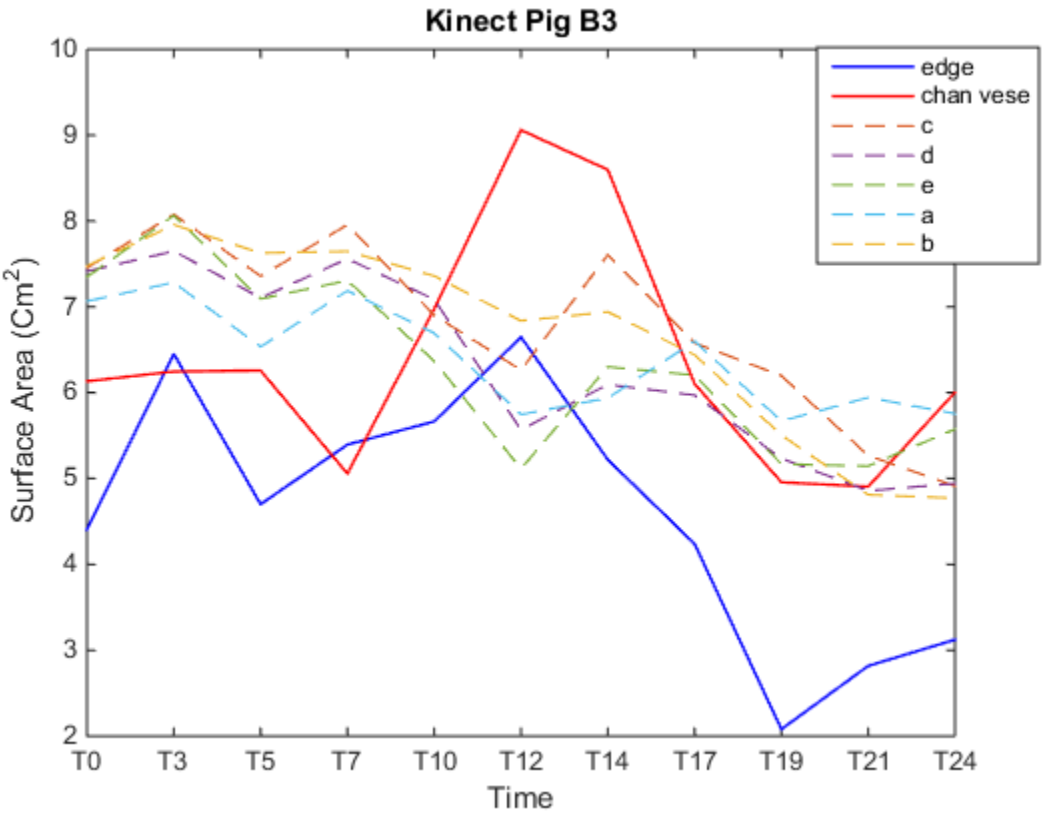

Kinect Pig A4
edge
chan vese
c
d
e
a
b
Surface Area (Cm2)
0
2
4
6
8
10
12
T0
T2
T3
T7
T9
T11
T14
T16
T18
T21
T25
T28
Time

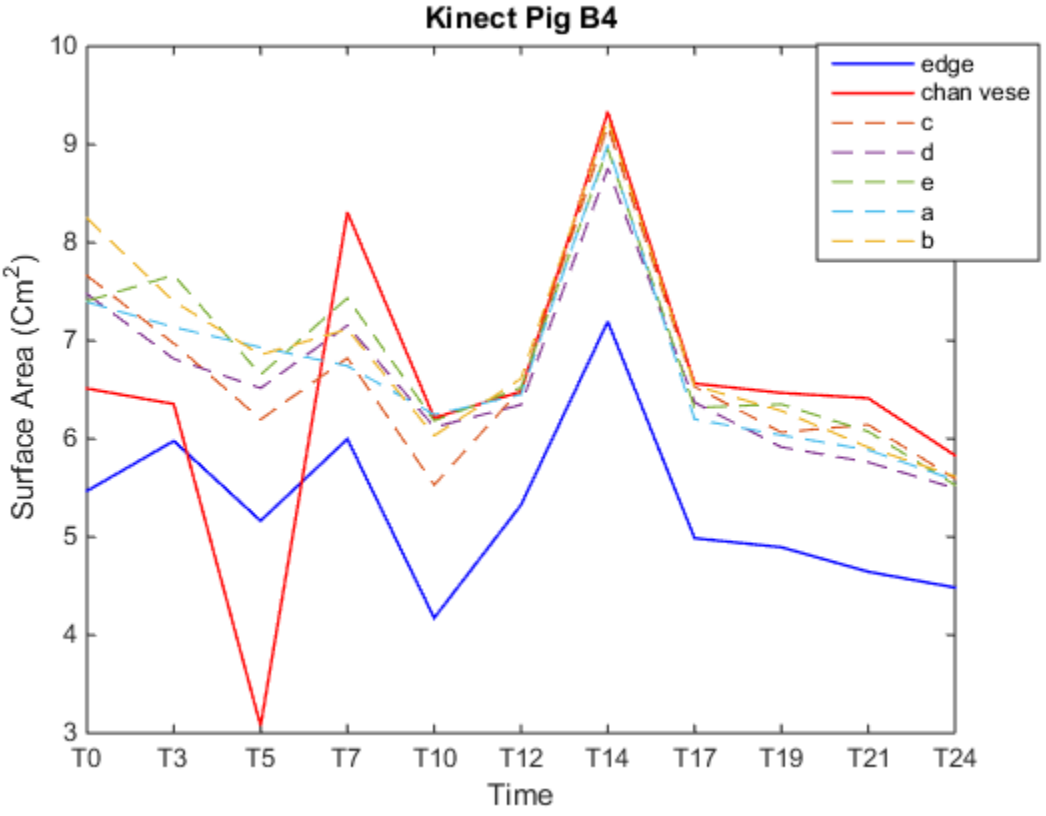

Kinect Pig A5
edge
chan vese
c
d
e
a
b
Surface Area (Cm2)
Time
T0
T2
T3
T7
T9
T11
T14
T16
T18
T21
T25
T28

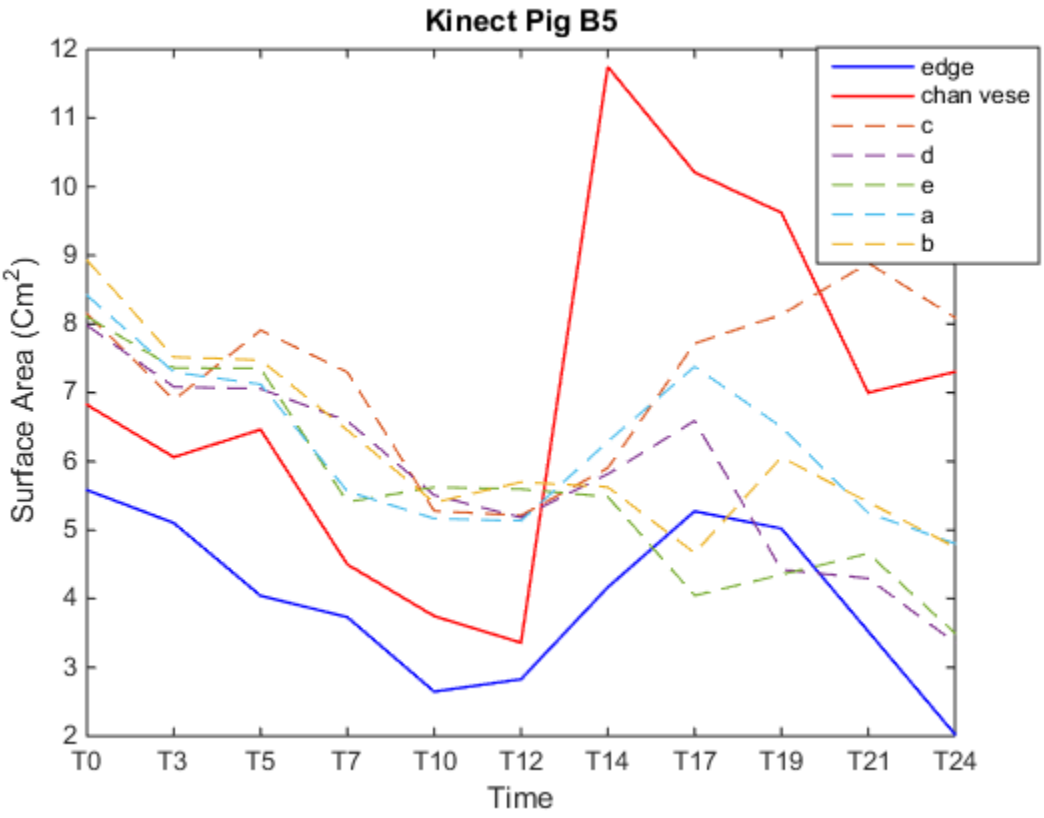

Kinect Pig A6
edge
chan vese
c
d
e
a
b
Surface Area (Cm2)
Time
T2
T3
T7
T9
T11
T14
T16
T18
T21
T25
T28

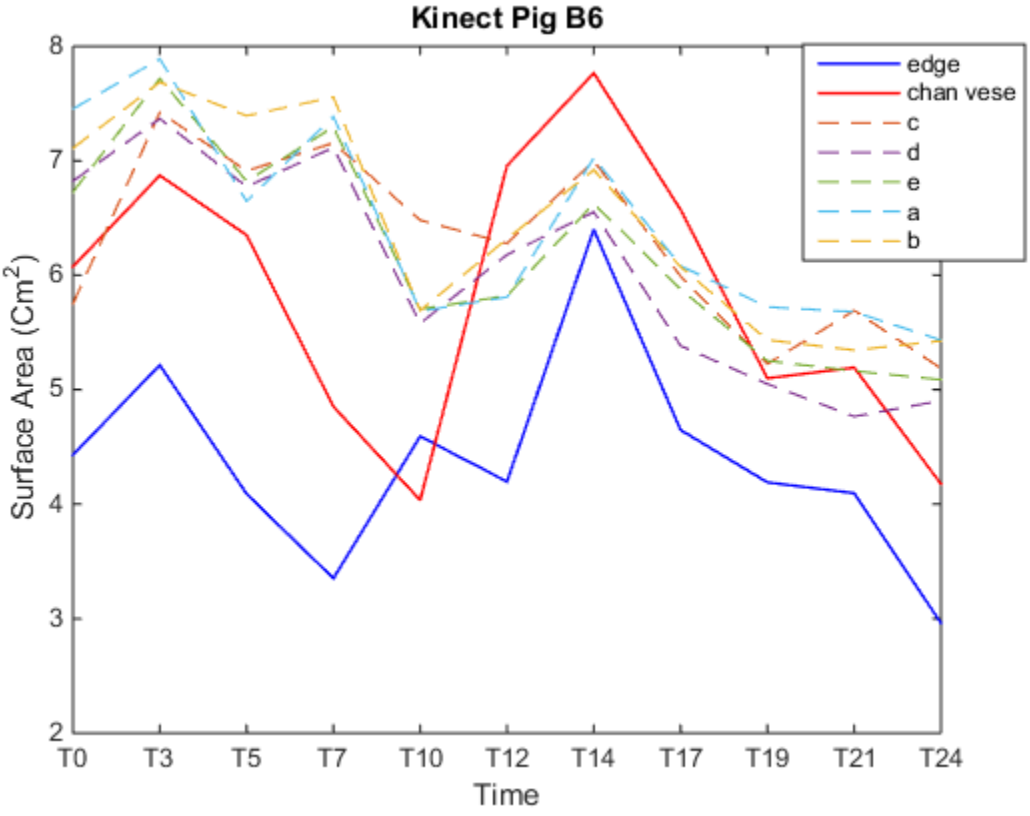

Kinect Pig B1
Surface Area (Cm2)
Time
edge
chan vese
c
d
e
a
b
T0
T3
T5
T7
T10
T12
T14
T17
T19
T21
T24


Kinect Pig B2
Surface Area (Cm2)
Time
edge
chan vese
c
d
e
a
b
T0
T3
T5
T7
T10
T12
T14
T17
T19
T21
T24


Kinect Pig B3
Surface Area (Cm2)
Time
edge
chan vese
c
d
e
a
b
10
9
8
7
6
5
4
3
2
T0
T3
T5
T7
T10
T12
T14
T17
T19
T21
T24


Kinect Pig B4
Surface Area (Cm2)
Time
edge
chan vese
c
d
e
a
b
10
9
8
7
6
5
4
3
T0
T3
T5
T7
T10
T12
T14
T17
T19
T21
T24


Kinect Pig B5
Surface Area (Cm2)
Time
edge
chan vese
c
d
e
a
b
T0
T3
T5
T7
T10
T12
T14
T17
T19
T21
T24


Kinect Pig B6
Surface Area (Cm2)
Time
edge
chan vese
c
d
e
a
b
T0
T3
T5
T7
T10
T12
T14
T17
T19
T21
T24

REFERENCES

[1] R. Papini, “Management of burn injuries of various depths.,” *BMJ*, vol. 329, no. 7458, pp. 158–60, 2004.

[2] S. Hettiaratchy and R. Papini, “Initial management of a major burn: II - Assessment and resuscitation,” *Bmj*, vol. 329, no. 7457, pp. 101–103, 2004.

[3] S. Hettiaratchy and P. Dziewulski, “Pathophysiology and types of burns,” *Br. Med. J.*, vol. 328, no. June, pp. 1427–1429, 2004.

[4] M. Kiser, G. Beijer, S. Mjuweni, A. Muyco, B. Cairns, and A. Charles, “Photographic assessment of burn wounds: A simple strategy in a resource-poor setting,” *Burns*, vol. 39, no. 1, pp. 155–161, 2013.

[5] T. L. Wachtel, C. C. Berry, E. E. Wachtel, and H. A. Frank, “The inter-rater reliability of estimating the size of burns from various burn area chart drawings,” *Burns*, vol. 26, no. 2, pp. 156–170, 2000.

[6] K. C. Lee, J. Dretzke, L. Grover, A. Logan, and N. Moiemen, “A systematic review of objective burn scar measurements,” *Burn. Trauma*, vol. 4, no. 1, p. 14, 2016.

[7] F. Kaliyadan, J. Manoj, S. Venkitakrishnan, and A. D. Dharmaratnam, “Basic digital photography in dermatology,” *Indian J Dermatol Venereol Leprol*, vol. 74, no. 5.

[8] Microsoft, “Kinect: Technical Specifications,” vol. 1, no. 2, pp. 3–5, 2014.

[9] V. P. Zharov, S. Ferguson, J. F. Eidt, P. C. Howard, L. M. Fink, and M. Waner, “Infrared Imaging of Subcutaneous Veins,” *Lasers Surg. Med.*, vol. 34, no. 1, pp. 56–61, 2004.

[10] E. Lachat, H. Macher, T. Landes, and P. Grussenmeyer, “Assessment and calibration of a RGB-D camera (Kinect v2 Sensor) towards a potential use for close-range 3D modeling,” *Remote Sens.*, vol. 7, no. 10, pp. 13070–13097, 2015.

[11] M. Hansard, S. Lee, O. Choi, and P. Horaud Radu, *Time of Flight Cameras: Principles, Methods, and Applications*. 2012.

[12] D. Pascale, “A review of RGB color spaces... from xyY to R’G’B’,” *Babel Color*, pp. 1–35, 2003.

[13] S. McHugh, “Understanding White Balance,” *Cambridge Color A Learn. Community Photogr.*, pp. 1–5, 2012.

[14] W.-H. Liu, Yung-Cheng Chan and Y.-Q. Chen, “Automatic White Balance for Digital Still Cameras,” *IEEE Trans. Consum. Electroincs*, vol. 41, no. 3, pp. 1–13, 1995.

[15] M. Kass, A. Witkin, and D. Terzopoulos, “Snakes: Active contour models,” *International Journal of Computer Vision*, vol. 1, no. 4. pp. 321–331, 1988.

[16] X. Qian, J. Wang, S. Guo, and Q. Li, “An active contour model for medical image segmentation with application to brain CT image.,” *Med. Phys.*, vol. 40, no. 2, p. 21911, 2013.

[17] N. Otsu, “A threshold selection method from gray-level histograms,” *IEEE Trans. Syst. Man. Cybern.*, vol. 9, no. 1, pp. 62–66, 1979.

[18] R. Crandall, “Image segmentation using the Chan-Vese algorithm,” *Proj. Rep. from ECE*, 2009.

[19] T. F. Chan and L. A. Vese, “Active contours without edges,” *IEEE Trans. Image Process.*, vol. 10, no. 2, pp. 266–277, 2001.

[20] E. Gedraite and M. Hadad, “Investigation on the effect of a Gaussian Blur in image filtering and segmentation,” *ELMAR, 2011 Proc.*, no. September, pp. 14–16, 2011.

[21] K. H. Zou *et al.*, “Statistical Validation of Image Segmentation Quality Based on a Spatial Overlap Index,” *Acad. Radiol.*, vol. 11, no. 2, pp. 178–189, 2004.

[22] P. A. Zijdenbos AP, Dawant BM, Margolin RA, “Morphometric analysis of white matter lesions in MR images: method and validation,” *Med. Imaging*, vol. 13, no. 4, pp. 716–724, 1994.

[23] K. Aainsqatsi, “Three degrees of burns.” 2007.