

TORSO SURFACE CONSTRUCTION, ELECTRODE LOCATION & HEART IMAGE REGISTRATION FOR INVERSE ELECTROCARDIOGRAPHY USING THE KINECT SENSOR

Author: Victor Sao Paulo Ruela
Supervisor: Dr. R. Martin Arthur

ESE 497 - Undergraduate Research
School of Engineering & Applied Science
Washington University in St. Louis
victorspruela@gmail.com

March 9, 2015

1 Introduction

1.1 Problem

The problem addressed by this work was to acquire torso-surface geometry, electrode location and heart-image registration data to enable torso-heart model construction for inverse electrocardiography using the Kinect sensor [2]. This work is based on the acquisition protocol developed by Jason Trobaugh using an Immersion 3DL mechanical digitizer [6].

1.2 Objective

The objective was to verify the feasibility of using the Kinect to generate a 3D model of a patient's torso and collect points of interest on the pictures taken to reduce the time taken and the cost of the protocol described in the Appendix D of Shuli Wang's dissertation [8]. This improvement would be accomplished by using the Kinect to collect all geometric data, drastically reducing the need of the 3D Immersion digitizer.

1.3 Resources

- MATLAB, Meshlab
- Microsoft XBOX 360 Kinect
- Vac-Lok patient immobilizing system with vaccum pump
- Biosemi body-surface mapping system
- 3D Immersion digitizer (3DL)
- Custom ultrasonic calibration phantom designed by Jason W. Trobaugh and built by Patrick T. Harkins [7]

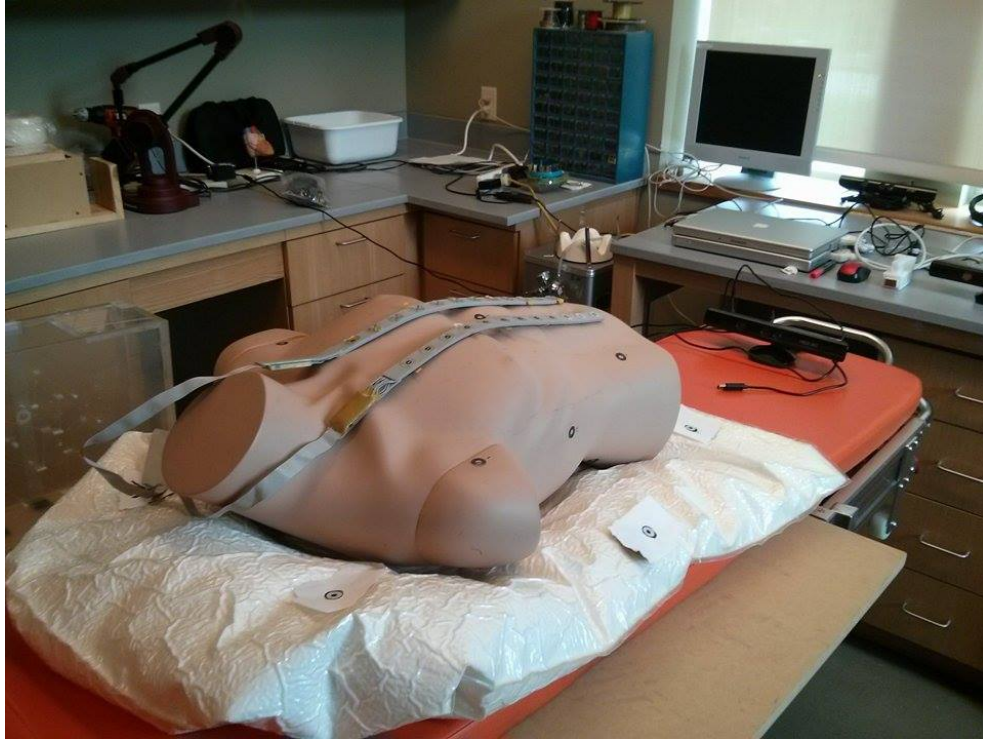


Figure 1: Equipment used for experiments

2 Protocol

The protocol for data acquisition is implemented by the file **patient_script.m**, described in the data acquisition and analysis step. The program structure can be seen in 2

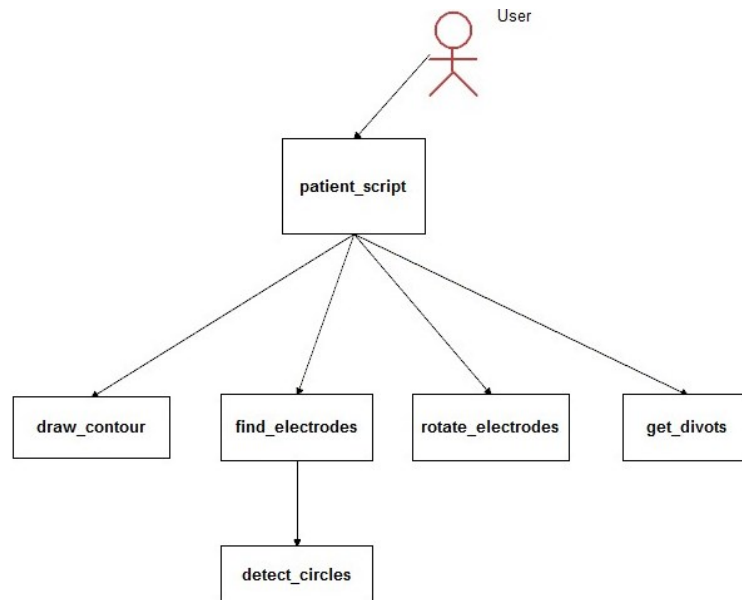


Figure 2: Relationship between the routines

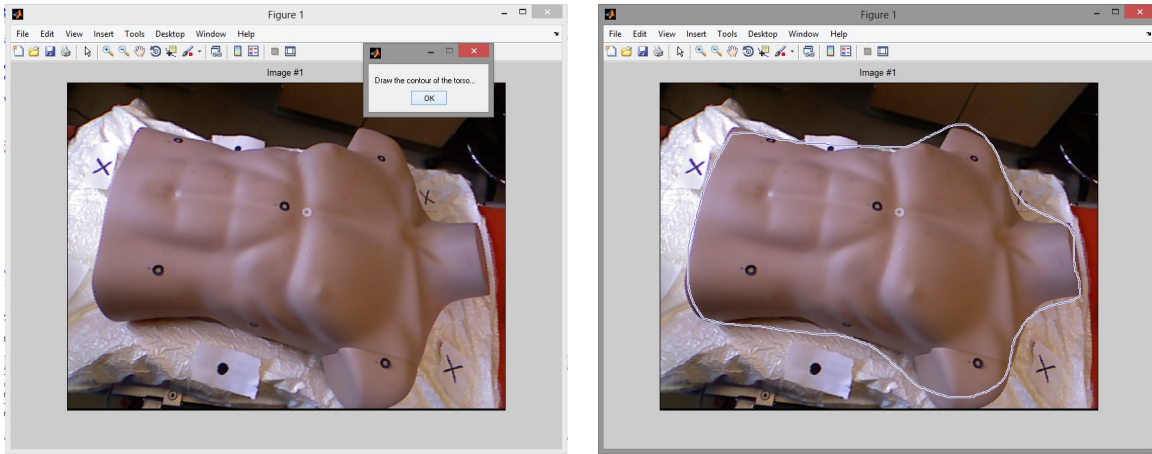
Moreover, it manages the profile of different patients, allowing to create new ones or load if

patient with ID inserted already exists. The ID consists of 6 characters and follows the pattern: [First name initial][Last name initial][Date in DD/MM]. For example, "vr0716" is a valid ID.

2.1 Routines

2.1.1 draw_contour.m

Asks user to draw with the mouse the contour of the torso on the color image provided. This is done with the function `imfreehand`. Then it generates a mask containing the region of interest.



(a) Torso contour drawing

(b) Torso contour drawing - continuation

Figure 3: `contour_draw.m` example

INPUT: image, idx, $N \times M \times 3$ and 1×1

OUTPUT: mask, $N \times M$

USAGE: `mask = draw_contour(image)`

2.1.2 detect_circles.m

Returns X,Y coordinates of electrodes from the color image provided. It is enhanced using histogram equalization with `adapthisteq`. Then `imfindcircles` detects the electrodes on the image. After that, it prompts the user to select the first and last electrodes of a strip which are used to interpolate a straight line between these two points. It selects the closest centers to the line. After this step it asks the user to remove centers that are not electrodes and to add new ones.

INPUT: color_im, $N \times M \times 3$

OUTPUT: centers, $K \times 2$ or -1 if canceled by user

USAGE: `centers = detect_circles(color_im)`

2.1.3 find_electrodes.m

Finds electrodes X,Y from the color image provided. This function simply asks the user if there any strips left to work on the image and calls **detect_circles** to identify the electrodes. This is necessary when we have multiple strips on a single image. It returns -1 when the user cancels the operation.

INPUT: image, $N \times M \times 3$

OUTPUT: centers, $K \times 2$

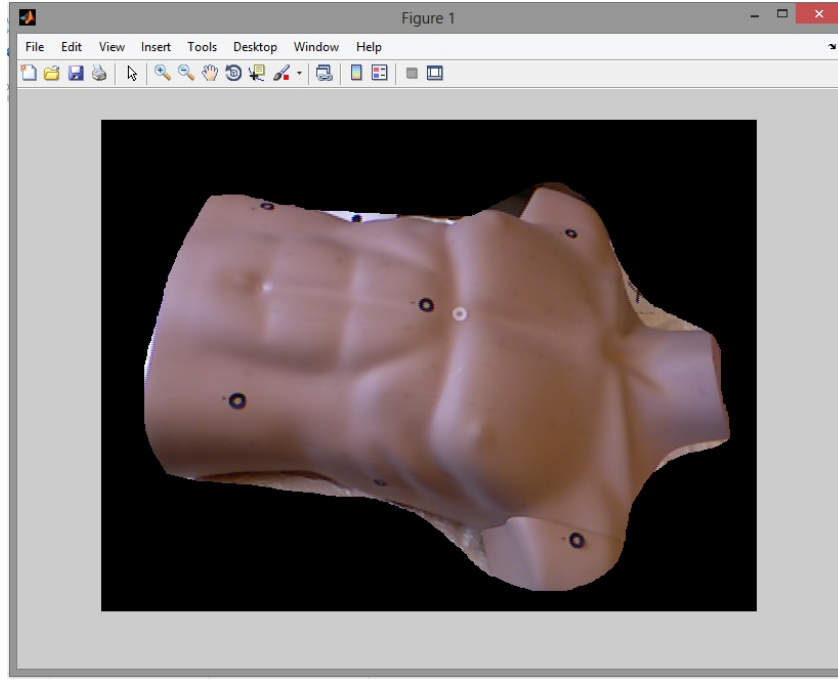


Figure 4: Mask applied to the color image

USAGE: centers = find_electrodes(image)

2.1.4 rotate_electrodes.m

Returns the X,Y and Z coordinates of the electrodes obtained from color image. It maps the provided centers to the equivalent value on x_{im} , y_{im} and $depth_{im}$. Then, it applies the homogeneous transformation with matrix T , placing the points in the correct coordinate system.

INPUT: $T, x_{im}, y_{im}, depth_{im}, centers$. $4 \times 4, N \times M, N \times M, N \times M, K \times 2$

OUTPUT: electrodes, $K \times 3$

USAGE: electrodes = rotate_electrodes($T, x_{im}, y_{im}, depth_{im}, centers$)

2.1.5 get_divots.m

Returns the X,Y and Z coordinates of the divots. The user manually selects the 5 divots on the image provided, following the order prompted by the routine.

INPUT: image, x_{im}, y_{im}, z_{im} $M \times N, M \times N, M \times N, M \times N$

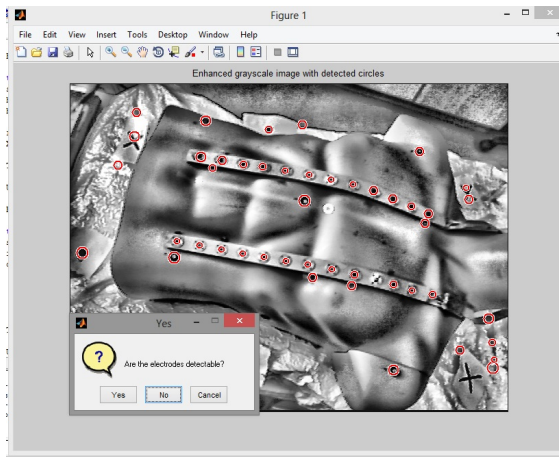
OUTPUT: divots, 5×3

USAGE: divots = get_divots(image, x_{im}, y_{im}, z_{im})

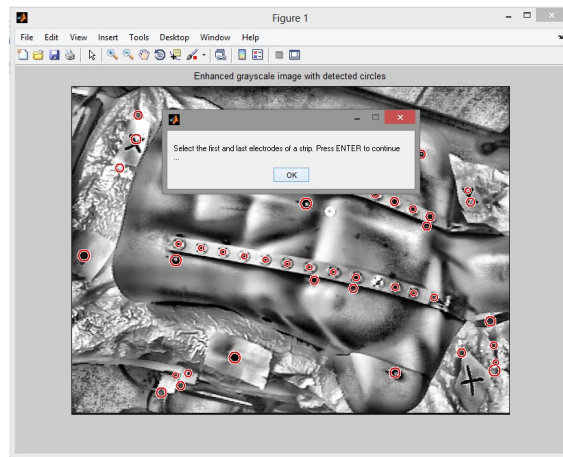
2.2 Data collection

In total, 7 pictures of the patient are taken. Before starting, the user should place the markers on the bean bag, which should be already pumped, as well as the fiducials on the torso. There should be a total of 6 markers: 2 on the top, one on each side and two on the bottom.

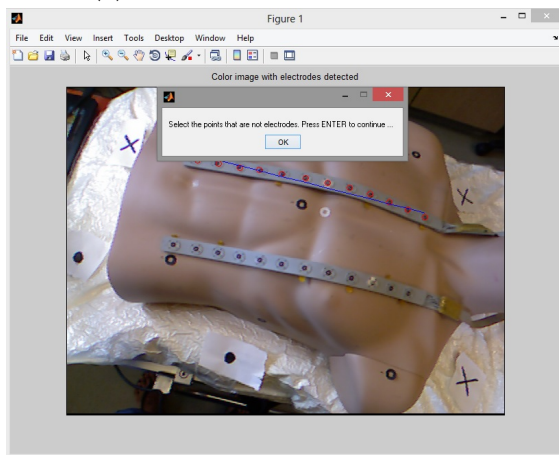
The program prints the instructions for the operator in the MATLAB console, so he knows the order and location in which the pictures should be taken with the Kinect, which are:



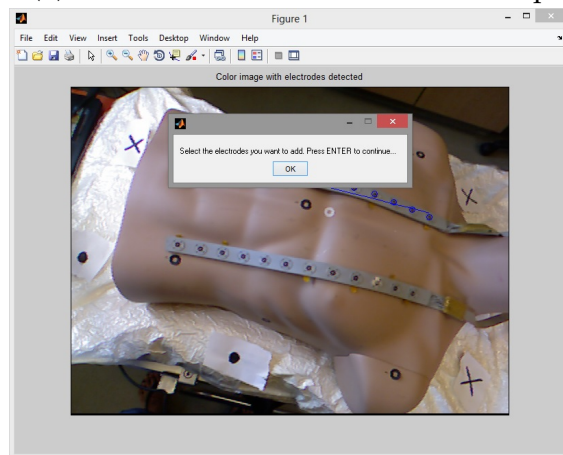
(a) Ask if electrodes are visible



(b) Select first and last electrodes of strip



(c) Remove points



(d) Add points

Figure 5: detect_circles.m example

1. Right side
2. Right side
3. Left side
4. Right side
5. Left side
6. Top

Picture 1 - Phantom This picture is necessary to obtain the position of the divots on the Kinect coordinate system. Some special procedures should be followed while taking this picture, in order to correctly generate the transformation matrix:

- Right bottom and side right bean bag markers should be visible;
- Top right, side right and bottom right fiducials on the torso should be visible;

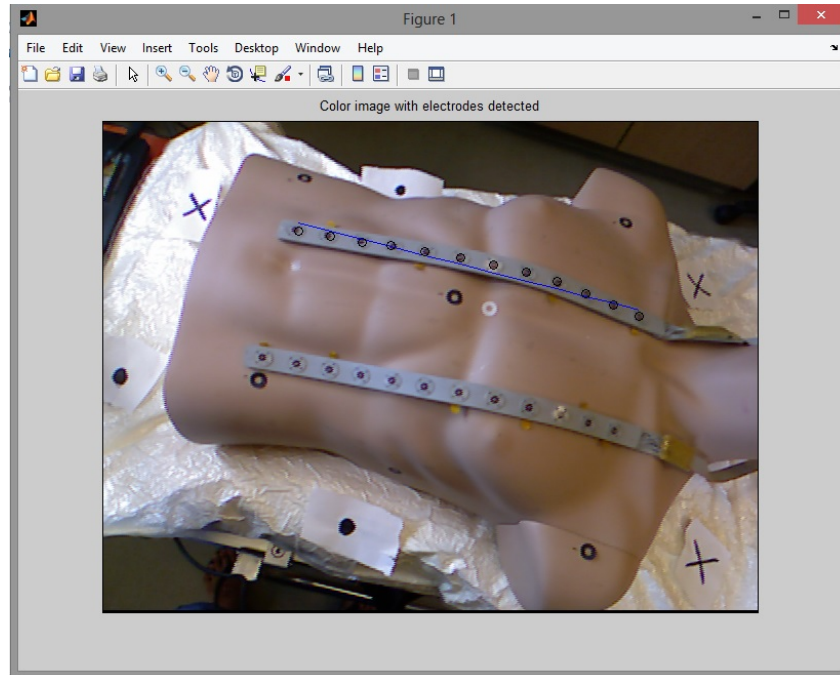


Figure 6: Electrodes detected in this image



Figure 7: Markers placed on bed

- Make sure the markers on the phantom are clearly visible and not very close to the borders of the image

I observed in the experiments that regions closer to the borders were sometimes being not correctly captured by the Kinect, resulting in a NaN value. This can affect the fiducials identification procedure which causes the homogeneous transformation to fail. Because of that, the last item should be extended to all the other pictures.

Pictures 2 and 3 - Torso without electrodes These 2 pictures are necessary to generate the front torso point cloud. Picture 2 is used as the base for the homogeneous transformation, so making sure that most of the fiducials and at least 3 markers are visible on this image is recommended.

Pictures 4,5 and 6 - Torso with electrodes on These 3 pictures are used to identify the electrodes. During the experiments, it was observed that the light was reflecting on the electrodes' metal part and making them look brighter. This affects their identification because the routine **detect_circles** relies on the fact that they are dark circles to work. So, while taking the pictures, the operator should avoid having any source of light that can cause this problem. Furthermore, to ensure that all electrodes are identified, the pictures of the torso's side should have the side electrode strip clearly visible. Picture 6 is an extra one taken from the top to make sure all strips are visible.

Picture 7 - Torso back impression This last picture is necessary to generate the back torso model. The operator should draw on the bean bag the torso's contour with a colored marker before asking the patient to get off the Vac-Lok. This is highly recommended because to remove extraneous points we use the **draw_contour** routine. Moreover, make sure all the 6 bean bag markers are visible on the image.

2.3 Analysis

2.3.1 Manually register fiducials

After all the data was collected, we start its analysis by manually registering the fiducials on the image. The program guides the operator through the whole process, asking the user to select the points in the correct order. Special attention must be paid in this step in order to reduce the mean correspondence error and prevent the homogeneous transformation to fail in the next step.

2.3.2 Calculate rigid transformations

Calculates the homogeneous transformation matrices for the fiducials and markers selected previously using the **findrigid.m** routine. If it fails for one of the images it means there are not enough points matching, therefore the user should redo last step and if the error persists, redo the data collection.

In this step, the mean correspondence error is also an output in the console. It depends on how close the user selected the center of the fiducials and markers in each image. A very high error means that the user incorrectly selected at least one of the fiducials in that image, therefore he should redo last section. The ideal scenario is a very small error, with a value ranging from 0 to 10. A big source of this error comes from the depth image because its values are integer variables, reducing the precision.

2.3.3 Transform point clouds

Applies the transformation matrices obtained previously to the respective point clouds. This is done to pictures 2, 3 and 7. The **draw_contour** routine is called to remove the extraneous points, by applying the mask to the respective depth images. This step is very important because the extraneous points can cause the 3D surface modeling to fail, especially the Radial Basis function. Therefore, the

user should pay attention to this step and draw the contour in a way that only the torso is included in the mask.

2.3.4 Apply transformation to divots

This step consists in:

1. Ask the user to select the divots on the image;
2. Match the selected divots' position with the ones from the 3DL;
3. Apply the homogeneous transformation to the transformed 3DL divots and wires location to place them in the main coordinate system.

The operator must choose the divots in the correct order and also carefully select them to reduce the mean correspondence error. This addition to the protocol in [8] eliminates the need of the 3D Immersion digitizer for the geometric data acquisition.

2.3.5 Detect electrodes of front images of torso

Calls the function **find_electrodes** to identify the electrodes on the torso.

2.3.6 Apply principal axis transformation to point cloud, electrodes and divots

This is the last step, consisting of calculating the principal axis transformation to the point cloud, electrodes and divots, plotting them to the user afterwards. Finally, all the data used in this step is saved to disk: point cloud as an ASCII file and electrodes and divots as .mat files. The following plot is shown (the axis were removed for better visualization):

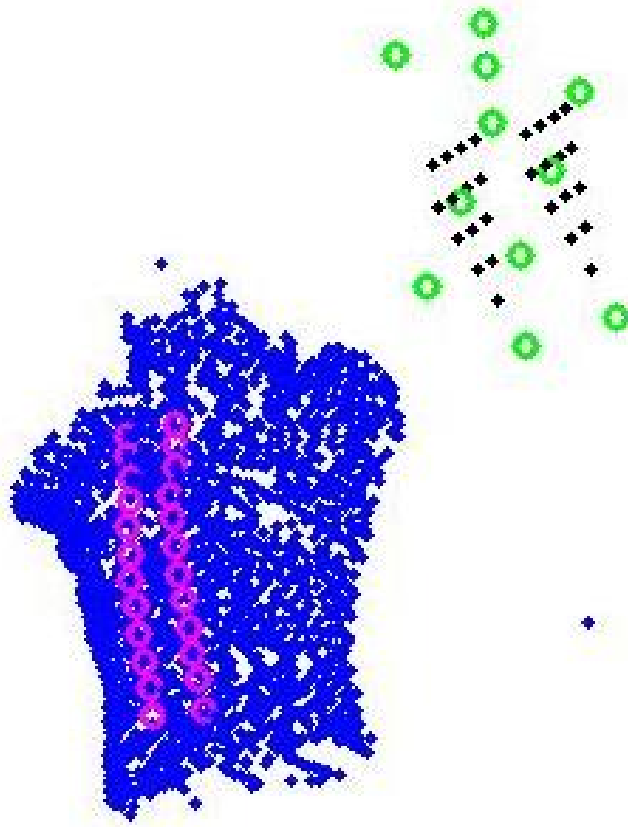


Figure 8: Point cloud (blue) with electrodes(magenta), divots (green) and phantom wires (black)

3 Discussion & Future Work

From the point cloud a 3D surface was modeled using two approaches: Poisson and Radial Basis function (RBF). The Poisson model was achieved with Meshlab and the RBF one was provided by Dr. R. Martin Arthur. Both yielded very good results (9 and 10). The Poisson model contains less detail but worked for almost all experiments. Moreover, it is not very sensitive to a few extraneous points, which may cause the RBF to fail. However, having a huge gap between the back and front of the torso can cause it to fail. Instead of generating a surface of the sides of the torso, the Poisson function finds a plane, making the model invalid. The RBF approach yields more detail for the torso surface, but is also very sensitive to extraneous points. For some experiments it failed for having only a few outliers. For this reason, manually removing all these points is suggested before using the RBF.

The time spent during the data collection was about 6 minutes. This includes the time needed for placing 2 electrodes strips on the mannequin. The experiment was conducted by only one person. This time can be reduced by having at least 2 people conducting the experiment: one to move around with the Kinect and another one in charge of the computer.

Future work includes identifying the electrodes' impression on the bean bag. Because the mannequin didn't have enough weight I could not register a good impression, making this task impossible to perform automatically. Another task is to integrate the routines necessary to register the ultrasound of the heart to `patient_script.m` and perform the experiment on a real person.

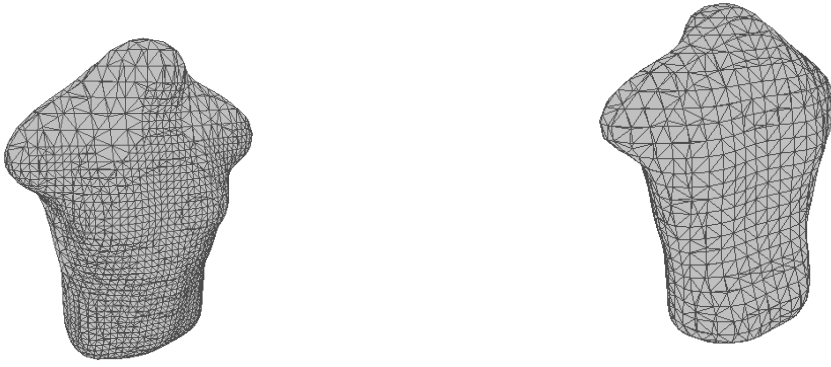
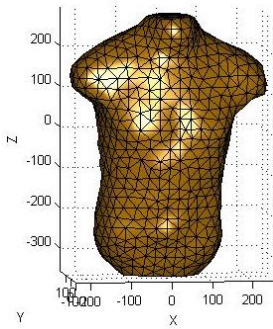
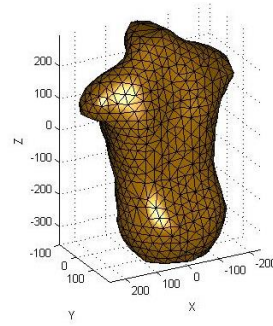


Figure 9: Torso surface model generated with Poisson



(a) Torso front



(b) Torso back

Figure 10: Torso surface model generated with the RBF

4 Conclusion

All the objectives proposed were met. The Kinect was fully integrated to the geometric data collection and the 3DL is no longer needed for this task. The electrodes can be automatically identified as well as the divots of the phantom. The 3DL was not completely eliminated because we still need to use it to track the transducer that collects the ultrasound data. The results are very promising, but we still need to perform experiments on real people. I believe it will work and turn out to be a viable alternative to acquire heart and torso geometry. Using Dr. Arthur's routine, the final result is shown in [11](#)

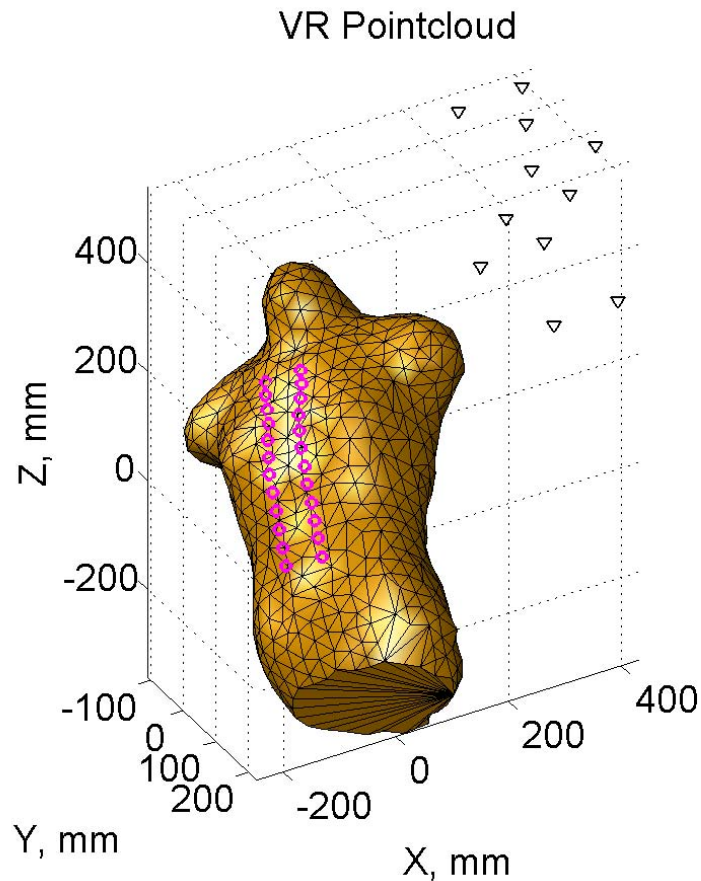


Figure 11: Final result

References

- [1] R. Martin Arthur and Jason W. Trobaugh. Electrocardiographic textbooks based on template hearts warped using ultrasonic images. *IEEE Transactions on Biomedical Engineering*, 59:2531,2537, September 2012.
- [2] K Khoshelham and SO Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454, 2012.
- [3] MathWorks. *adapthisteq*. Online. Accessed on 10-July-2014.
- [4] MathWorks. *imfindcircles*. Online. Accessed on 10-July-2014.
- [5] Steven L. Eddins Rafael C. Gonzalez, Richard E. Woods. *Digital Image Processing Using MATLAB*. Gatesmark, second edition, 2009.
- [6] JW Trobaugh. patientscript.m. Personal Communication, 2006.
- [7] JW Trobaugh and RM Arthur. Methods for using ultrasound to generate a heart surface for electrocardiographic inverse problems. *Int J of Bioelectromagnetism*, 5:314–315, 2003.
- [8] Shuli Wang. *Electrocardiographic consequences of electrical and anatomical remodeling in diabetic and obese humans*. PhD thesis, Washington University in St. Louis, 2009.

Appendix A - How to set up Kinect

The following steps worked in a machine running Windows 8 64-bit and should work for others versions. The Kinect must be connected to a USB 3.0 port.

- Download and install 'OPENNI-Win32-1.0.0.23.exe'
- Plug in the Kinect.
- Download and extract the PrimeSensor modules for OpenNI from <https://github.com/avin2/SensorKinect/tree/master>
- Download and install 'SensorKinect-Win32-5.00.exe'
- Open the Device Manager and look for the Kinect. Right click on it and select 'Update Driver Software'. Go to Browse my computer for driver manually - Pick from a list of device drivers on my computer - Have disk. Go to folder /Platform/Win32/Drivers and select the '.inf' file. Follow the instructions and finish.
- The step above should be done to the Kinect camera and motor. The audio driver is not needed. A 'PrimeSense' device should appear now on the list.
- Open NiViewer.exe demo program to verify that everything worked (it should be in C:/Program Files/OpenNI/Sample/Bin/Release)

To collect the images, the user needs a 32-bit S.O or a 32-bit MATLAB version. **openni_mex.mexw32** should be in the same folder as the **patient_script.m**. **video_capture.m** routine should be run to verify if the data collection is working.

5 Appendix B - Matlab Code

5.1 draw_contour.m

```
% DRAW_CONTOUR select region of color image
% Washington University in St. Louis
% Dept of Electrical Engineering
% Victor Ruela
% FILE: draw_contour.m
% DESCRIPTION: asks user to draw the contour of the torso in the color
% image using the mouse.
% INPUT: image, idx, NxMx3 and 1x1
% OUTPUT: mask, NxM
%
% USAGE: centers = detect_circles(color_im)
```

```
function mask = draw_contour(image,idx)
```

```
    imshow(image);
```

```
title(['Image #', num2str(idx)]);  
uiwait(msgbox('Draw the contour of the torso...'));  
hFH = imfreehand();  
mask = hFH.createMask();
```

```
end
```


5.2 detect_circles.m

```
% DETECT_CIRCLES detect circles in color image.
%   Washington University in St. Louis
%   Dept of Electrical Engineering
%   Victor Ruela
%   FILE: detect_circles.m
%   DESCRIPTION: Find X,Y coordinates of electrodes from color
%   image. Image is enhanced using histogram equalization then
%   imfindcircles is used to detect the electrodes. The first and last
%   electrodes of a strip are selected and a straight line is interpolated. The
%   closest centers detected are selected. The user has the option to
%   add/remove any electrodes detected after this step.
%   INPUT: color_im, NxMx3
%   OUTPUT: centers, Kx4 or -1 if canceled by user
%
%   USAGE: centers = detect_circles(color_im)

function centers = detect_circles(color_im)

%%% constants declaration %%%
sensitivity = 0.87;
tiles = [12 12];
clip_limit = 0.1;
radius_range = [4,8];
d_threshold = 15;
%%%          -----          %%%

% convert to grayscale
gray = rgb2gray(color_im);
% apply histogram equalization enhancement
j = adapthisteq(gray, 'NumTiles', tiles, 'ClipLimit', clip_limit);
imshow(j);
% find the circles in the enhanced image
[c1,r1] = imfindcircles(j, radius_range, 'ObjectPolarity', 'Dark', 'Sensitivity' ...
    , sensitivity , 'Method', 'TwoStage');
viscircles(c1,r1);
title('Enhanced grayscale image with detected circles');

% ask user to verify if electrodes are detectable
choice = questdlg('Are the electrodes detectable?', 'Yes', 'No');
if(strcmp(choice, 'No'))
    close all; centers = []; return;
end
% checks if user canceled. Returns -1 for this case
if strcmp(choice, 'Cancel')
    disp('User cancelled...'); close all; centers = -1; return;
end

% ask user to select first and last electrodes of strip
uiwait(msgbox('Select the first and last electrodes of a strip. Press ENTER to continue ...'));

[ex , ey] = ginput(2);
```

```

m = (ey(2) - ey(1))/(ex(2) - ex(1));
b = ey(1) - m*ex(1);

% add the selected electrodes to the centers
%c1 = [c1 ; [ex ey]];
% sort centers in ascending order of the X coordinate
c1 = sort_centers(c1);

% remove the points with x<min(ex) and x>max(ex)

idxs = find( ~(round(c1(:,1)) < min(ex) | round(c1(:,1)) > max(ex)) );
ct1 = c1(min(idxs):max(idxs),1);
ct2 = c1(min(idxs):max(idxs),2);
c1 = [ct1 ct2];

% calculate distance of points from the polynomial

d = [];

for k=1: numel(c1(:,1))

    dist = abs( c1(k,2) - m*c1(k,1) - b )/sqrt(m^2 +1);
    d(k) = dist;

end
% get rid of points far from the polynomial

cnew = [];

for k=1: numel(d)

    if(d(k) < d_threshold) %threshold for this case, can change for another pictures
        cnew = [cnew; c1(k,1) c1(k,2)];
    end

end

% show image with centers detected and straight line interpolated

close all;
imshow(color_im);
hold on;
scatter(cnew(:,1),cnew(:,2), 'r');
c = cnew(:,1);
d = m*c + b;

plot(c,d);
title('Color image with electrodes detected');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% start user interaction
% asks user to select points that are not electrodes
uiwait(msgbox('Select the points that are not electrodes. Press ENTER to continue ...'));
[x,y] = getpts;

```

```

% find the corresponding centers

c_temp = cnew;

% calculate distance from points selected to other points
for i=1:size(x)
    d = [];
    c_temp_new = [];
    c_temp_new1 = [];
    c_temp_new2 = [];

    for k=1:size(c_temp(:,1))
        dist = sqrt((c_temp(k,1)-x(i))^2 + (c_temp(k,2)-y(i))^2);
        d(k) = dist;
    end
    % select closest point index
    idx = find(d == min(d));

    % remove this point from the vector
    if idx > 1
        if idx == size(c_temp(:,1)) % point is in the last position
            c_temp_new(:,1) = c_temp(1:idx-1,1);
            c_temp_new(:,2) = c_temp(1:idx-1,2);
        else
            c_temp_new1(:,1) = c_temp(1:idx-1,1);
            c_temp_new1(:,2) = c_temp(1:idx-1,2);

            c_temp_new2(:,1) = c_temp(idx+1:end,1);
            c_temp_new2(:,2) = c_temp(idx+1:end,2);

            c_temp_new = [c_temp_new1 ; c_temp_new2];
        end
    else % point is in the first position
        c_temp_new(:,1) = c_temp(idx+1:end,1);
        c_temp_new(:,2) = c_temp(idx+1:end,2);
    end
    c_temp = c_temp_new;
end

hold on
scatter(c_temp(:,1),c_temp(:,2),'b');

% asks user to select points that are electrodes
uiwait(msgbox('Select the electrodes you want to add. Press ENTER to continue...'));

[x2,y2] = getpts;

if ~isempty(x2)
    c_temp = [c_temp ; [x2 y2]];
end

hold on
scatter(c_temp(:,1),c_temp(:,2),'k');

```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
centers = c_temp;
```

```
end
```

```
% Sort centers according to x coordinate using Bubble sort algorithm
```

```
function x = sort_centers(x)
```

```
n = length(x(:,1));
```

```
while (n > 0)
```

```
    % Iterate through x
```

```
    nnew = 0;
```

```
    for i = 2:n
```

```
        % Swap elements in wrong order
```

```
        if (x(i,1) < x(i - 1,1))
```

```
            x = swap(x,i,i - 1);
```

```
            nnew = i;
```

```
        end
```

```
    end
```

```
    n = nnew;
```

```
end
```

```
end
```

```
function x = swap(x,i,j)
```

```
% Swap center(i) and center(j)
```

```
val = [x(i,1) x(i,2)];
```

```
x(i,1) = x(j,1);
```

```
x(i,2) = x(j,2);
```

```
x(j,1) = val(1);
```

```
x(j,2) = val(2);
```

```
end
```

5.3 find_electrodes.m

```
% FIND_ELECTRODES find electrodes location from color image.
%   Washington University in St. Louis
%   Dept of Electrical Engineering
%   Victor Ruela
%   FILE: find_electrodes.m.
%   DESCRIPTION: Find electrodes indices from color image.
%   Keeps asking user if there are any electrodes strips left and then calls
%   function detect_circles to identify the electrodes.
%   INPUT: image, NxMx3
%   OUTPUT: centers, Kx2
%
%   USAGE: centers = find_electrodes(image)

function centers = find_electrodes(image)

centers = [];
choice = 'Yes';

while strcmp(choice, 'Yes')

    xy = [];

    xy = detect_circles(image);
% checks if user canceled function detect_circles
    if xy == -1
        disp('User cancelled...');close all;return;
    end
% asks if user wishes to continue working with this image
    choice = questdlg('Are there any strips left?', 'Yes', 'No');
% checks if user canceled
    if strcmp(choice, 'Cancel')
        disp('User cancelled...');close all;return;
    end
    centers = [centers; xy];

end
% checks if electrodes were detected in this image
if isempty(centers)
    disp('No electrodes detected...');close all;return;
end

% plots electrodes that were detected
close all;
imshow(image);
hold on;
scatter(centers(:,1),centers(:,2));
title('Color image with all electrodes');

end
```

5.4 rotate_electrodes.m

```
% ROTATE_ELECTRODES apply homogeneous transformation to points.
%   Washington University in St. Louis
%   Dept of Electrical Engineering
%   Victor Ruela
%   FILE: rotate_electrodes.
%   DESCRIPTION: Finds electrodes X,Y,Z coordinates from indices obtained in
%   color image. Applies homogeneous transformation with transformation matrix T.
%   imfindcircles is used to detect the electrodes.
%   INPUT: T,x_im,y_im,depth_im,centers. 4x4,NxM,NxM,NxM,Kx2
%   OUTPUT: electrodes, Kx3
%
%   USAGE: electrodes = rotate_electrodes(T,x_im,y_im,depth_im,centers)

function electrodes = rotate_electrodes(T,x_im,y_im,depth_im,centers)

% check if the centers variable is not empty
if isempty(centers)
    disp('Electrodes were not detected in this image');electrodes=[];return;
end

zc = [];

    for k=1:max(size(centers))

        x = x_im(round(centers(k,2)),round(centers(k,1)));
        y = y_im(round(centers(k,2)),round(centers(k,1)));
        z = depth_im(round(centers(k,2)),round(centers(k,1)));

        zc = [zc; x y z];

    end

    electrodes = zc';
    electrodes(4,:) = 1;
    electrodes = T* electrodes;
    electrodes = electrodes(1:3,:)'

end
```


5.5 get_divots.m

```
% GET_DIVOTS find divots location on color image
%   Washington University in St. Louis
%   Dept of Electrical Engineering
%   Victor Ruela
%   FILE: get_divots.m.
%   DESCRIPTION: Find divots X,Y and Z coordinates from color image.
%
%   INPUT: image,x_im,y_im,z_im MxN,MxN,MxN,MxN
%   OUTPUT: divots, 5x3
%
%   USAGE: divots = get_divots(image,x_im,y_im,z_im)

function divots = get_divots(image,x_im,y_im,z_im)

imshow(image);
divots = [];
divots_order = {'top left';'top right';'center';'bottom left';'bottom right'};

for j = 1:numel(divots_order)

    title(['Select ', divots_order{j}, ' divot']);
    [in_x,in_y] = ginput(1);
    x = x_im(round(in_y), round(in_x));
    y = y_im(round(in_y), round(in_x));
    z = z_im(round(in_y), round(in_x));
    divots = [divots; x y z];

end

end
```

6 patient_script.m

```
% PATIENT_SCRIPT - high-level script for evaluating a single patient
% revised 8/22/03 & 10/06 by Jason W. Trobaugh
% Modified by R. Martin Arthur 7/12 to use kinect pair
% Modified by Victor Ruela 6/25 to intergrate kinect pictures

% Before starting the data collection:
% Follow instructions on D.2 section - Preparation

clear all; close all; clc;
%% Constants

% Depth threshold to remove background.
thresh = 1000;

% Number of Kinect images to use.
num_ims_eon = 3; num_back_ims = 1; num_front_ims = 2;
num_ims = num_ims_eon + num_back_ims + num_front_ims;

% Number of pictures to be taken with kinect
n = 100;

% Labels of bean bag fiducials.
marker_labels = {'top left'; 'top right'; 'side left'; 'side right'; 'bottom left'; ...
    'bottom right'};
num_markers = numel(marker_labels);

% Labels of mannequin fiducials.
% man_labels={'front top left'; 'front top right'; 'side left'; 'front center'; 'side right'; ...
% 'front bottom left'; 'front bottom right'; 'back top left'; 'back top right'; ...
% 'back center left'; 'back center right'; 'back bottom left'; 'back bottom right'};
% num_man = numel(man_labels);
% man_labels={'front top left'; 'front top right'; 'front center'; ...
% 'front bottom left'; 'front bottom right'};
man_labels={'front top left'; 'front top right'; 'side left'; 'front center'; 'side right'; ...
    'front bottom left'; 'front bottom right'};
num_man = numel(man_labels);

picture_directions = {'righth side', 'left side', 'righth side', 'left side', 'top'};

%% Header: create new patient profile

% Input patient ID (last init., first init., 4 char. date, e.g. cb0822)
ID = input('Patient ID: ', 's');
DataDir = ['C:\Users\Victor Ruela\Desktop\Research\data_acquisition_protocol\' , ID, '\'];

% create new patient if ID doesn't exist
if (~exist(DataDir, 'dir')); mkdir(DataDir); end
```

```

%% Start data acquisition protocol (can change)
% 1) Attach reference electrodes and back and right side electrodes to the subject
% 2) Run ActiView software and check for signal quality
% 3) Place patient on Vac-Lock cushion
% 4) Mark 3 table references 9(?) and 4 anatomic references points ...
% (right arm, left arm, left leg and CMS electrode sites)
% 5) Check the quality and ranges of the back electrodes signals
%% 5.1) Take pictures of patient without electrodes on
tStart = tic;

save_file = 'kinect_data.mat';
if exist([DataDir,save_file], 'file')
    disp(['Loading Kinect data from ', save_file, '...']);
    load([DataDir,save_file]);
else
%%
    openni_mex('init');
    clc;
    % Take picture of the phantom
    disp('Take the picture farther enough so the fiducials');
    disp('and divots are not too close to the borders.');
```

```

        choice = 'Yes';
        while strcmp(choice, 'Yes')

            disp('Press the SPACE key to record a Kinect image...'); pause;
            disp('Saving image...');
            for j=1:n
                % 'Grab' to take a snapshot of the data
                openni_mex('grab');

                % Access the image and depth from the previous grab
                image=openni_mex('image');
                depth=openni_mex('depth');
```

```

        end;
        imshow(image);
        choice = questdlg('Would you like to take another picture?', 'Yes', 'No');

        if strcmp(choice, 'Cancel')
            disp('User canceled taking pictures. Returning...'); close all; return;
        end

    end

phantom_ims = image;
[X_phantom, Y_phantom, Z_phantom] = fix_units(depth);

% Take pictures of patient without torso electrodes on

X_ims = cell(num_ims,1);
Y_ims = cell(num_ims,1);
Z_ims = cell(num_ims,1);
color_ims = cell(num_ims,1);

% First one: left side

```

```

% Second one: righth side
% Make sure that all 5 markers are visible in pictures, they are needed to
% generate the transformation matrixes
    openni_mex('init');
    clc;
    disp('Taking 2 pictures with electrodes off');
    for ii=1:(num_front_ims)

        choice = 'Yes';
        while strcmp(choice, 'Yes')

            disp(['Place the camera on ', picture_directions{ii}, ' of torso']);
            disp('Make sure that at least 4 markers are visible');
            disp('and they are not very close to the borders');
            disp('Press the SPACE key to record a Kinect image...'); pause;
            disp('Saving image...');
            for j=1:n
                % 'Grab' to take a snapshot of the data
                openni_mex('grab');

                % Access the image and depth from the previous grab
                image=openni_mex('image');
                depth=openni_mex('depth');

            end;
            imshow(image);
            choice = questdlg('Would you like to take another picture?', 'Yes', 'No');

            if strcmp(choice, 'Cancel')
                disp('User canceled taking pictures. Returning...'); close all; return;
            end

        end

        color_ims{ii} = image;
        [X_ims{ii}, Y_ims{ii}, Z_ims{ii}] = fix_units(depth);
        close all;
        clear image; clear depth;
    end

    % Finished two pictures of the patient without electrodes on

end
%%
% 6) Acquire ultrasound images (skip for now)
% 7) Attach front electrodes
% 8) Check the quality and range of front electrodes
% 9) Record body-surface ECGs
%% 10) Collect torso surface data with kinect

% Get Kinect Data
save_file = 'kinect_data.mat';
if exist([DataDir, save_file], 'file')
    %disp(['Loading Kinect data from ', save_file, '...']);
    % load([DataDir, save_file]);
else

```

```

%%
% Take the three first three pictures with electrodes on
% First one: left side
% Second one: right side
% Third one: top of torso
clc;
disp('Taking 3 pictures with electrodes on');

%openni_mex('init');
for ii=4:(4+num_ims_eon-1)

    choice = 'Yes';
while strcmp(choice,'Yes')

    disp(['Place the camera on ',picture_directions{ii-1}, ' of torso']);
    disp('Make sure that at least 4 markers are visible');
    disp('and they are not very close to the borders');
    disp('Press the SPACE key to record a Kinect image...'); pause;
    disp('Saving image...');
    for j=1:n
        % 'Grab' to take a snapshot of the data
        openni_mex('grab');

        % Access the image and depth from the previous grab
        image=openni_mex('image');
        depth=openni_mex('depth');

    end;
    imshow(image);
    choice = questdlg('Would you like to take another picture?','Yes','No');

    if strcmp(choice,'Cancel')
        disp('User canceled taking pictures. Returning...');close all;return;
    end

end

    color_ims{ii} = image;
    [X_ims{ii},Y_ims{ii},Z_ims{ii}] = fix_units(depth);
    close all;
    clear image;clear depth;
end

% Finished collecting the three pictures of the patient with electrodes on
clc;
disp('Draw the position of torso sides in the Vac-Lok');
disp('Remove front electrodes from patient')
disp('Help patient sit up and get off the Vac-Lok');
disp('Take picture of the patient impression on the Vac-Lok');

    choice = 'Yes';
while strcmp(choice,'Yes')

    disp('Place the camera on top of torso');
    disp('Make sure that at least 4 markers are visible');
    disp('and they are not very close to the borders');

```

```

disp('Press the SPACE key to record a Kinect image...'); pause;
disp('Saving image...');
for j=1:n
    % 'Grab' to take a snapshot of the data
    openni_mex('grab');

    % Access the image and depth from the previous grab
    image=openni_mex('image');
    depth=openni_mex('depth');

end;
imshow(image);
choice = questdlg('Would you like to take another picture?','Yes','No');

if strcmp(choice,'Cancel')
    disp('User canceled taking pictures. Returning...');close all;return;
end

end
color_ims{3} = image;
[X_ims{3},Y_ims{3},Z_ims{3}] = fix_units(depth);
imshow(color_ims{3});
close all;
clear image;clear depth;
% Finished taking the Vac-Lok impression picture. Shut down kinect.
openni_mex('shutdown');
% Save data collected
save([DataDir,save_file],'X_ims','Y_ims','Z_ims','color_ims','phantom_ims',...
    'X_phantom','Y_phantom','Z_phantom');
disp(['Saving Kinect data to ', save_file, '...']);
end

tElapsed = toc(tStart);
%% Manually register fiducials
crop = nan(4,num_ims+1);
fiducials = nan(num_markers,3,num_ims + 1);
man_fiducials = nan(num_man,3,num_ims + 1);
save_file = 'registration_bean.mat';

if exist([DataDir,save_file], 'file')
    disp(['Loading fiducial registration data from ', save_file, '...']);
    load([DataDir,save_file]);
else
    %%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % start working with the phantom figure
    % first, select the divots on the image
    figure;
    x_im = X_phantom;
    y_im = Y_phantom;
    depth_im = Z_phantom;
    color_im = phantom_ims;

    %setnan = depth_im>thresh | isnan(Z_phantom);
    setnan = isnan(Z_phantom);

```



```

color_im(repmat(setnan, [1, 1, 3])) = nan;
depth_im(setnan) = nan;
imagesc(color_im);
colormap gray
axis off
truesize

idx = num_ims + 1;

title('Select top left corner of crop.')
top_left = round(ginput(1));
crop(1,idx) = top_left(1);
crop(2,idx) = top_left(2);

title('Select bottom right corner of crop.')
bottom_right = round(ginput(1));
crop(3,idx) = bottom_right(1);
crop(4,idx) = bottom_right(2);
cropped_im = color_im(round(crop(2,idx)):round(crop(4,idx)),...
    round(crop(1,idx)):round(crop(3,idx)),:);
cropped_depth = depth_im(round(crop(2,idx)):round(crop(4,idx)),...
    round(crop(1,idx)):round(crop(3,idx)));
cropped_X = x_im(round(crop(2,idx)):round(crop(4,idx)),...
    round(crop(1,idx)):round(crop(3,idx)));
cropped_Y = y_im(round(crop(2,idx)):round(crop(4,idx)),...
    round(crop(1,idx)):round(crop(3,idx)));

imagesc(cropped_im);
colormap gray
axis off
truesize
set(gcf, 'units', 'normalized', 'outerposition', [0 0 1 1]);
crop_image_size = size(cropped_im);
for j = 1:num_markers
    title(['Select the ', marker_labels{j}, ' bean bag fiducial.'],...
        'Click outside of the image if it is not visible.')}

    [in_x, in_y] = ginput(1);
    if ~(in_y < 0 || in_y > crop_image_size(1) || ...
        in_x < 0 || in_x > crop_image_size(2))
        x = cropped_X(round(in_y), round(in_x));
        y = cropped_Y(round(in_y), round(in_x));
        z = cropped_depth(round(in_y), round(in_x));
        fiducials(j,:,idx) = [x y z];
    end
end

for j = 1:num_man
    title(['Select the ', man_labels{j}, ' mannequin fiducial.'],...
        'Click outside of the image if it is not visible.')}

    [in_x, in_y] = ginput(1);
    if ~(in_y < 0 || in_y > crop_image_size(1) || ...
        in_x < 0 || in_x > crop_image_size(2))
        x = cropped_X(round(in_y), round(in_x));
        y = cropped_Y(round(in_y), round(in_x));

```

```

        z = cropped_depth(round(in_y), round(in_x));
        man_fiducials(j,:,idx) = [x y z];
    end
end

title('Press any key to continue.')
pause;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Work with the other pictures
figure
for idx = 1:num_ims
    x_im = X_ims{idx};
    y_im = Y_ims{idx};
    depth_im = Z_ims{idx};
    color_im = color_ims{idx};
    setnan = depth_im>thresh | isnan(depth_im);
    %setnan = isnan(Z_phantom);
    color_im(repmat(setnan, [1, 1, 3])) = nan;
    depth_im(setnan) = nan;
    imagesc(color_im);
    colormap gray
    axis off
    truesize

    title('Select top left corner of crop.')
    top_left = round(ginput(1));
    crop(1,idx) = top_left(1);
    crop(2,idx) = top_left(2);

    title('Select bottom right corner of crop.')
    bottom_right = round(ginput(1));
    crop(3,idx) = bottom_right(1);
    crop(4,idx) = bottom_right(2);
    cropped_im = color_im(round(crop(2,idx)):round(crop(4,idx)),...
        round(crop(1,idx)):round(crop(3,idx)),:);
    cropped_depth = depth_im(round(crop(2,idx)):round(crop(4,idx)),...
        round(crop(1,idx)):round(crop(3,idx)));
    cropped_X = x_im(round(crop(2,idx)):round(crop(4,idx)),...
        round(crop(1,idx)):round(crop(3,idx)));
    cropped_Y = y_im(round(crop(2,idx)):round(crop(4,idx)),...
        round(crop(1,idx)):round(crop(3,idx)));

    imagesc(cropped_im);
    colormap gray
    axis off
    truesize
    set(gcf, 'units', 'normalized', 'outerposition', [0 0 1 1]);
    crop_image_size = size(cropped_im);
    for j = 1:num_markers
        title(['Select the ', marker_labels{j}, ' bean bag fiducial.'],...
            'Click outside of the image if it is not visible.')}

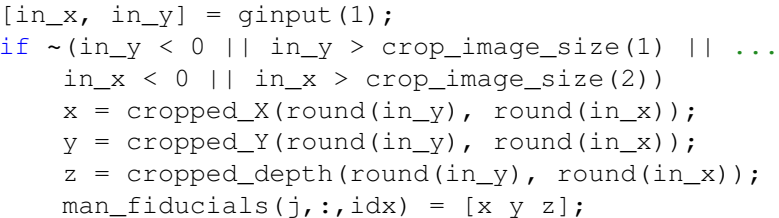
        [in_x, in_y] = ginput(1);
        if ~(in_y < 0 || in_y > crop_image_size(1) || ...
            in_x < 0 || in_x > crop_image_size(2))

```

```

        x = cropped_X(round(in_y), round(in_x));
        y = cropped_Y(round(in_y), round(in_x));
        z = cropped_depth(round(in_y), round(in_x));
        fiducials(j,:,idx) = [x y z];
    end
end

for j = 1:num_man
    title(['Select the ', man_labels{j}, ' mannequin fiducial.'],...
          'Click outside of the image if it is not visible.')
```



```

    [in_x, in_y] = ginput(1);
    if ~(in_y < 0 || in_y > crop_image_size(1) || ...
        in_x < 0 || in_x > crop_image_size(2))
        x = cropped_X(round(in_y), round(in_x));
        y = cropped_Y(round(in_y), round(in_x));
        z = cropped_depth(round(in_y), round(in_x));
        man_fiducials(j,:,idx) = [x y z];
    end
end

title('Press any key to continue.')
pause;
end

disp(['Saving fiducial registration data to ', save_file, '...']);
save([DataDir,save_file],'crop','fiducials','man_fiducials')
end

%% Calculate Rigid Transformations (Rotation and Translation) Using Beanbag Fiducials
% From current image to previous image. (All images points should be in the
% same coordinate system after this.)
% Note: At least three points must be shared between each neighboring
% image.

transformations = cell(1,num_ims);
ref_idx = 1;
transformations{ref_idx} = eye(4);
i = 2;
wbar = waitbar(0, 'Finding Transformations');
while i <= num_ims
    idx = mod(i,num_ims);
    if idx == 0; idx = num_ims; end;
    % target = fiducials(:, :, ref_idx)';
    % source = fiducials(:, :, idx)';
    target = [fiducials(:, :, ref_idx)' man_fiducials(:, :, ref_idx)'];
    source = [fiducials(:, :, idx)' man_fiducials(:, :, idx)'];

    matches = ~isnan(source(1,:)) & ~isnan(target(1,:)) &...
              ~isnan(source(2,:)) & ~isnan(target(2,:)) &...
              ~isnan(source(3,:)) & ~isnan(target(3,:));
    % target (prev) = T * source (cur)
    T = findrigidrma(target(:,matches), source(:,matches));
    % Apply previous transform as well to map to same coordinates.
    T = transformations{ref_idx} * T;
    transformations{idx} = T;
end

```

```

    i = i+1;
    waitbar(i/(num_ims),wbar);
end

idx = num_ims+1;
% Calculate transformation for phantom image
target = [fiducials(:, :, ref_idx)' man_fiducials(:, :, ref_idx)'];
source = [fiducials(:, :, idx)' man_fiducials(:, :, idx)'];
matches = ~isnan(source(1, :)) & ~isnan(target(1, :));

T = findrigidrma(target(:, matches), source(:, matches));
% Apply previous transform as well to map to same coordinates.
T = transformations{ref_idx} * T;
transformations{idx} = T;

waitbar(i/(num_ims),wbar);
close(wbar);

%% Plot Transformed Fiducials

figure
hold all
for idx = 1:num_ims+1
    points = squeeze(fiducials(:, :, idx))';
    % Apply homogeneous transform...
    points(4, :) = 1;
    points = transformations{idx} * points;
    points = points';
    % Matching fiducials have the same color.
    plot3(points(:, 1), points(:, 2), points(:, 3), '.', 'markersize', 5);
end

title('Transformed fiducials'); xlabel('X'); ylabel('Y'), zlabel('Z');
hold off
axis vis3d; pause; close all;

%% Transform Point Clouds
% Transform point cloud only for front images without electrodes on and
% the back one. They are the only ones needed to generate it

point_clouds = cell(1, num_back_ims + num_front_ims);

for idx = 1:(num_back_ims + num_front_ims)
    disp(['Aligning image ', num2str(idx), '...']);
    x_im = X_ims{idx};
    y_im = Y_ims{idx};
    color_im = color_ims{idx};
    depth_im = Z_ims{idx};
    setnan = depth_im > thresh | isnan(depth_im);
    %setnan = isnan(depth_im);
    % Asks user to draw contour of the image
    choice = 'Yes';
    while strcmp(choice, 'Yes')
        mask = draw_contour(color_im, idx);
    end
end

```

```

choice = questdlg('Would you like to draw the coutour again?','Yes','No');

if strcmp(choice,'Cancel')
    disp('User canceled torso contour drawing');close all;return;
end

end

depth_im(~mask | setnan) = nan;

cropped_x = x_im(round(crop(2,idx)):round(crop(4,idx)),...
    round(crop(1,idx)):round(crop(3,idx)));
cropped_y = y_im(round(crop(2,idx)):round(crop(4,idx)),...
    round(crop(1,idx)):round(crop(3,idx)));
cropped_depth = depth_im(round(crop(2,idx)):round(crop(4,idx)),...
    round(crop(1,idx)):round(crop(3,idx)));
cropped_color = color_im(round(crop(2,idx)):round(crop(4,idx)),...
    round(crop(1,idx)):round(crop(3,idx)),:);
cropped_size = size(cropped_depth);

% Crop each depth image...
x = reshape(cropped_x,[1 prod(cropped_size)]);
y = reshape(cropped_y,[1 prod(cropped_size)]);
z = reshape(cropped_depth,[1 prod(cropped_size)]);
close all;

good_points = ~isnan(x) & ~isnan(y) & ~isnan(z);
% Homogeneous transform...
point_clouds{idx} = ones(4,numel(find(good_points)));
point_clouds{idx}(1:3,:) = [x(good_points); y(good_points); z(good_points)];
% Apply transformation to all points to map them to the same coordinate
% system.
point_clouds{idx} = transformations{idx} * point_clouds{idx};
point_clouds{idx} = point_clouds{idx}(1:3,:);

end

%% Detect electrodes on impression on the Vac-Lok
electrodes_back = [];

%% Remove electrodes impression from point cloud

%% Combine Point Cloud
cloud_x = []; cloud_y = []; cloud_z = [];

for idx = 1:(num_front_ims + num_back_ims)
    cloud_x = [cloud_x ; point_clouds{idx}(1,:)'];
    cloud_y = [cloud_y ; point_clouds{idx}(2,:)'];
    cloud_z = [cloud_z ; point_clouds{idx}(3,:)'];
end

%% Transform the Divots
load PhantomMeasurementsHome;
% First, match the divots from the picture with the ones stored in
% DivotsHome

```

```

divots = get_divots(phantom_ims,X_phantom,Y_phantom,Z_phantom);

T = findrigidrma(divots',DivotsHome(1:3,1:5));

PhantomDivots = T*DivotsHome;

PhantomWires = WiresHome;
PhantomWires(4,:) = 1;
PhantomWires = T*PhantomWires;

% Plot transformed divots

plot3v(PhantomDivots(1:3,1:5)', 'mo');
hold on
plot3v(divots, 'ko');
xlabel('X'); ylabel('Y'); zlabel('Z');
title('3DL -> Kinect aux ');
legend('3DL', 'Kinect');
saveas(gcf, [DataDir, '3DL-Kinect.fig']);
% axis image;grid on;
% cameramenu;camproj('orthographic');

% Now, apply the transformation matrix obtained previously to place them in
% the main reference system

PhantomDivots = transformations{num_ims + 1}*PhantomDivots;
divots = divots';
divots(4,:) = 1;
divots = transformations{num_ims + 1}*divots;

PhantomWires = transformations{num_ims + 1}*PhantomWires;

% Plot trasformed divots, now in the main coordinate system
figure;
plot3v([PhantomDivots(1:3,:)';PhantomWires(1:3,:)'], 'bo');
hold on;
plot3v(divots(1:3,:) ', 'ro');

title('Kinect aux -> Kinect base');
legend('3DL', 'Kinect');
xlabel('X'); ylabel('Y'); zlabel('Z');
saveas(gcf, [DataDir, 'Kinect1-Kinect2.fig']);
pause;
disp('Press any key to continue...'); close all;

%% Detect electrodes of front images of torso
electrodes = [];
for idx = 4:num_ims

% find the center of the electrodes in the color image
c_temp = find_electrodes(color_ims{idx});
% find the xyz coordinates for the electrodes from the centers detected
e_temp = rotate_electrodes(transformations{idx},X_ims{idx},Y_ims{idx},Z_ims{idx},c_temp);

electrodes = [electrodes ; e_temp];

```



```

end

electrodes = [electrodes ; electrodes_back];
pause;
disp('Press any key to continue...');
close all;
%% Apply principal axis transformation to point cloud, electrodes and divots

xyz=[cloud_x cloud_y cloud_z];
xyz = xyz(1:128:end,:);

divots = [PhantomDivots(1:3,:)'];
wires = [PhantomWires(1:3,:)'];
% apply principal axis transformation to point cloud
[pam,y,cy,mnx]=prinax(xyz);
% Make x(RL)-->-y, y(BF)-->z, z(DU)-->x
tmp=y; y(:,1)=-tmp(:,2); y(:,2)=tmp(:,3); y(:,3)=-tmp(:,1); y(:,1) = -y(:,1);

clear tmp;
% apply principal axis transformation to electrodes
electrodes_aux=(pam*(electrodes' - mnx(:,ones(1,length(electrodes))))')';
tmp=electrodes_aux; electrodes_aux(:,1)=-tmp(:,2); electrodes_aux(:,2)=tmp(:,3);...
    electrodes_aux(:,3)=-tmp(:,1);electrodes_aux(:,1) = -electrodes_aux(:,1);

clear tmp;
% apply principal axis transformation to divots
divots_aux=(pam*(divots' - mnx(:,ones(1,length(divots))))')';
tmp=divots_aux; divots_aux(:,1)=-tmp(:,2); divots_aux(:,2)=tmp(:,3); ...
    divots_aux(:,3)=-tmp(:,1); divots_aux(:,1) = -divots_aux(:,1);

clear tmp;
% apply principal axis transformation to wires
wires_aux=(pam*(wires' - mnx(:,ones(1,length(wires))))')';
tmp=wires_aux; wires_aux(:,1)=tmp(:,2); wires_aux(:,2)=tmp(:,3); ...
    wires_aux(:,3)=-tmp(:,1);

% plot the point cloud
plot3v(y, '.'); grid on; axis image;
% plot the electrodes on the point cloud
hold on;
plot3(electrodes_aux(:,1),electrodes_aux(:,2),electrodes_aux(:,3), 'mo', 'LineWidth',2.5);
hold on;
plot3(divots_aux(:,1),divots_aux(:,2),divots_aux(:,3), 'go', 'LineWidth',2.5);
hold on;
plot3(wires_aux(:,1),wires_aux(:,2),wires_aux(:,3), 'k.', 'LineWidth',2);
xlabel('X'); ylabel('Y'); zlabel('Z');
title('Mannequin via 3 Kinect Images using findrigidrma');
cameramenu; camproj('orthographic');
saveas(gcf, [DataDir, 'point-cloud.fig']);

% Save point cloud to ASCII file

```

```

filename = 'point_cloud.asc';
disp(['Writing data to ', filename]);
save([DataDir,filename], 'y', '-ascii');

% Save electrodes
filename = 'electrodes';
disp(['Writing data to ', filename]);
save([DataDir,filename], 'electrodes_aux');

% Save divots
divots_aux = [divots_aux;wires_aux];
filename = 'divots';
disp(['Writing data to ', filename]);
save([DataDir,filename], 'divots_aux');

%%
% 11) Clean up patient

```