Skin Lesion Surface Area and Erythema Intensity Calculation for PASI Determination using Stereo Vision

L.M. Jonker
Bachelor Thesis Committee:
dr. B. Sirmaçek, dr.ir. M. Abayazid, prof.dr.ir W. Steenbergen.
Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS), Robotics and Mechatronics (RaM), University of Twente, 7500 AE Enschede, The Netherlands.

July 3, 2019

***Abstract*—There are several tools to determine the severity of psoriasis and to track the effect of treatment, one of which is Psoriasis Area and Severity Index (PASI). PASI score calculation needs an accurate lesion area to total skin area ratio. To accurately determine these areas, 3D reconstruction from stereo vision has been used. Disparity maps and consecutively point clouds of the object of interest were made from stereo pictures. The object for testing this method has been the arm. Transforming the colour space from red, green and blue (RGB) to YCbCr (Y:luminance; Cb: chrominance-blue; Cr: chrominance-red) and inspecting the point clouds of each separate component, provides detection of red and lighter lesions. A point cloud of these lesions can then be made. Using Delaunay triangulation on the point clouds results in a mesh of which the area can be calculated. Tests have been conducted on light and dark skin shades, with red and pink lesions. It has also been tested for multiple lesions placed a few centimetres apart. The test results showed that the formulated method works on both light and dark shade skin for red lesions, and detection of multiple lesions can be done as well. The method does not perform as well on lighter, pink lesions. The calculated lesion areas differed to the actual areas with an average factor of +37.45%.**

I. Introduction

Surface reconstruction is widely used for applications such as environment mapping, 3D modelling, and medical imaging. In medical applications, surface reconstruction is used to inspect and calculate the surface area of the skin, providing knowledge when treating certain diseases. For psoriasis - which is mainly recognisable by red patches on the skin - knowing the skin area is an important aspect for determining the Psoriasis Area and Severity Index (PASI) score, which indicates the severity of the lesions and serves as an indication on how well the treatment works [1]. Determining this area is still a tough task, and doing so by hand takes a lot of time. An existing method which computes this score, uses a mono camera [2]. It enables doctors to quickly obtain the surface area and also determines the severity of the other criteria needed to calculate the PASI score. Other existing methods for surface reconstruction use mono cameras as well, or methods such as photogrammetry, laser scanning, or millimeter wave scanning. With these methods, the surface area of the skin can be calculated to diagnose, monitor, or to keep track of treatments [3]. The existing surface reconstruction methods which do not use pictures, do not compute a full coloured mesh. This makes detecting different coloured areas - such as psoriasis lesions - challenging. Furthermore, using mono cameras does not provide area calculation in three dimensions, possibly making the results less accurate than they could be. Existing methods also tend to be quite expensive, e.g. the reconstruction system at Robotics and Mechatronics (RaM) which costs over €70,000. Large camera setups could also make patients uncomfortable, which should be prevented if possible. This paper presents an affordable and handheld solution to surface calculation using stereo vision. As application, the method has been developed to detect lesions on different skin colours and calculate their area, making it useful for determining PASI scores. The lesion detection also determines the intensity of the redness.

II. Materials

The following materials have been used for implementation and testing of this method:

- Stereo Camera: *Non Distortion Dual Lens USB3.0 Camera Module Synchronization HD 960P OTG UVC Plug Play Driverless 3D VR Stereo Webcam* (€75). Outputs stereo images with a resolution of 960x2560, i.e. 960x1280 for each separate left and right image.
- Software: MATLAB R2019a, including the Image Processing Toolbox.
- Lesion samples: red and pink stickers have been made, with areas of 12 cm^2 and 6 cm^2. The accuracy of the surface calculation will be determined by comparing this area to the calculated area.

III. Methods

The PASI score is calculated by looking at four criteria per body part (head, arms, trunk and legs) [1], [2]:

- Area: the percentage of skin covered by the lesions, categorised as (0): 0%, (1): $1-9\%$, (2): $10-29\%$, (3): $30-49\%$, (4): $50-69\%$, (5): $70-89\%$, (6): $90-100\%$.
- Erythema: the redness of the lesions.
- Induration: the thickness of the lesions.
- Desquamation: the scaling of the lesions.

The bottom three criteria are categorised by five intensity scores: (0): Absent, (1): Mild, (2): Moderate, (3): Severe, (4): Very severe. This research will only present a solution to the first two criteria, the area and erythema of the lesion. These criteria can be determined by looking at the 3D model made from the stereo images. To make this 3D model, the process as seen in Figure 1 has been followed.

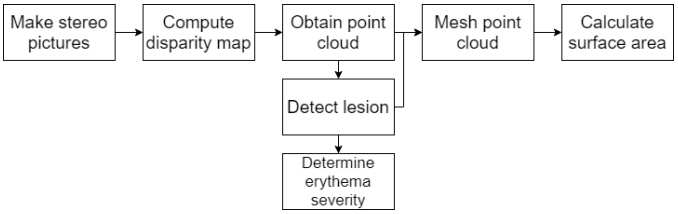


Fig. 1: Functional block diagram of the general process.

A. Make stereo pictures

Before being able to process stereo pictures, the camera has to be calibrated. This has been done with the Stereo Camera Calibrator app from MATLAB. After this, stereo pictures were made of an arm on which a lesion can be seen. For the tests, the lesion samples as described in Section II were examined.

B. Compute disparity map

From the taken stereo pictures, disparity maps were made by using MATLAB's Semi-Global Matching (SGM) function. SGM works especially well for fine structured objects and their borders and still works well under radiometric changes [4], [5]. It also provides less occlusions than Block Matching (BM), MATLAB's other disparity function. Disparity maps show the difference in distance between the left and right images; objects closer to the camera will have larger disparity than objects far away. Before being able to compute the disparity map, the disparity range should be determined. This has been done by drawing a line on the anaglyph image of each stereo pair, after which the range was automatically calculated. Doing so provides an accurate disparity range, such that it is correct for every stereo pair.

C. Obtain point cloud

Obtaining a correct, denoised point cloud takes a few steps. The disparity map should be correct, and not contain too many occlusions or noise, as the same noise present in the disparity map will be present in the point cloud as well. Removing the noise has been done by first down sampling the point cloud, then by looking at the distance between neighbouring points. If this distance is deemed too large, a point is considered noise. Another method is looking at the clusters, which are groups of points gathered in a certain area. Assuming the cluster of the arm is the largest one, smaller clusters are left out, leaving a denoised point cloud.

D. Mesh point cloud

For meshing the obtained volume point cloud, Delaunay triangulation has been used. The convex hull of the point cloud, or rather the mesh, in the *x*-*y* plane, consists of triangles formed by Delaunay edges [6]. Then the z coordinates of each point can be added, making the mesh three dimensional. However, using this method results in excess area, which should be left out when calculating the area of the arm and the lesion.

E. Calculate surface area

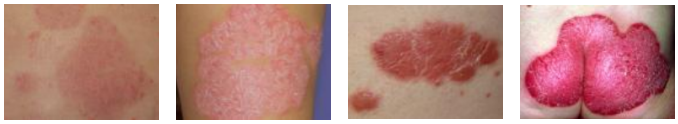
Since the mesh only consists of triangles, calculating the surface areas has been done by summing the area of the separate triangles. However, the triangulation method used here also makes triangles between outer points which are on opposite sides, since it is a convex hull. To solve this issue, the excess areas were detected by looking at their area and maximum edge length. These areas were not part of the summation.

F. Detect lesion

Using the YCbCr (Y: luminance; Cb: chrominance-blue; Cr: chrominance-red [7]) colour space deemed more reliable than the red, green and blue (RGB) colour space for detecting the lesions. By looking at an image in only the Cr component, the red areas - which have the highest Cr value - are highlighted. For lighter lesions the other two colours (Y and Cb) have been inspected as well. Since the area of the lesion should be determined from a 3D mesh, the lesion has been detected from the point cloud in the colour of one of the YCbCr components. A point cloud of only the lesion is left, which can be meshed and examined for area.

G. Determine erythema severity (redness)

The erythema severity of the lesion has been determined by comparing the redness of the stereo picture to the redness of the pictures as seen in Figure 2. To determine the redness, the Y, Cb, or Cr values are looked at, corresponding to which colour was used to detect the lesion. The colour values of the lesions were compared to those of the reference images. The severity is then the number of the picture the lesion corresponds to the most.



(a) Intensity 1. (b) Intensity 2. (c) Intensity 3. (d) Intensity 4.

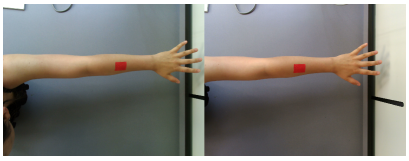
Fig. 2: Erythema intensity reference pictures [1].

H. Testing

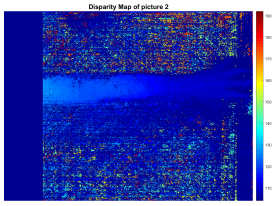
To test the method, tests have been conducted on different skin colours, using the red and pink stickers as mentioned in Section II. Additionally, the method has been tested for two separated lesions. The calculated areas were compared to the actual measured areas of the stickers by calculating the difference with Equation 1. For each test, four images were taken from a distance of approximately 1 meter.

$$\frac{|area_{\text{real}} - area_{\text{calculated}}|}{area_{\text{real}}} \cdot 100\%. \tag{1}$$

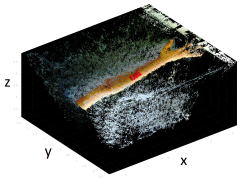
The PASI score has been determined for every test, using Equation 2. Assuming the sticker lesion is the only lesion on the whole body, the components regarding the head, trunk and legs are equal to zero and can be left out of the original equation [1].



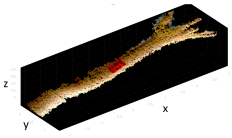
(a) Stereo image.



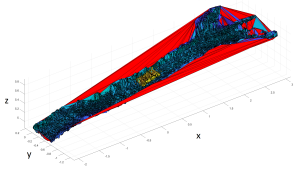
(b) Disparity map.



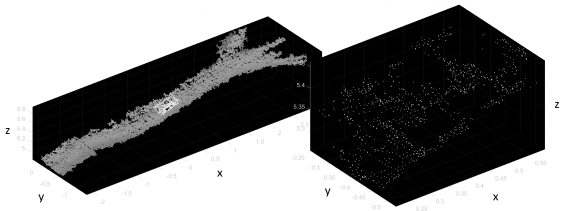
(c) Point cloud.



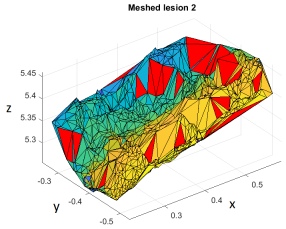
(d) Filtered point cloud.



(e) Mesh of the arm.



(f) Point clouds of arm and detected lesion in Cr.



(g) Detected, filtered lesion point cloud.

Fig. 3: Results of test with red lesion on light skin tone, stereo pair 2.

$$PASI = (arms_{\text{erythema}} \cdot 0.2) \cdot arms_{\text{area}}, \tag{2}$$

where $arms_{\text{erythema}}$ and $arms_{\text{area}}$ are the intensity scores mentioned at the beginning of this section. To calculate the lesion to arm area ratio, the calculated arm area has been multiplied by four, to represent both arms from both front and back.

IV. Results

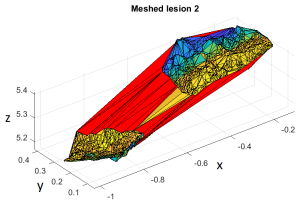
The meshes presented in this section have red triangles in them. These triangles were areas not taken into account when calculating the total surface area. They have been automatically detected. The axes in the 3D models are defined as follows: x: the length, y: the width, z: the height/depth of the arm/lesion. Note that the axes are in decimeters.

For the first test with one red lesion on light skin, the end results as well as the intermediate results of stereo pair 2 can be seen in Figure 3. For the other tests, per skin tone and lesion colour, only the stereo image and surface mesh of the lesions of the test with the best result (the highlighted tests in Table I) are shown. The calculated areas and the difference to the actual area of all tests can be seen in Table I. The calculated areas of the arm, detected erythema intensities and the calculated PASI scores can also be found here, as well as the used colour components which yielded the best lesion detection. The visual results of the other tests, i.e. surface meshes and point clouds are presented in Appendix A.

The disparity map in Figure 3b has a lot of noise in the background area, and so does the point cloud in Figure 3c. The detected lesion in Figure 3f seems to have holes in it. This can be seen again in the mesh in Figure 3g. The mesh in this figure also seems to have a lot of texture, different from the used stickers.

The results presented in Table I show that the method has some difference for every test. The smallest average difference is +4.18%, while the largest average difference is +141.86%. The average of all average differences is +37.45%. Notable is that the test with smallest difference of -0.47% (Figure 3), has many wrongly detected excess areas. Furthermore, the mesh of which the shape and detected excess areas closely resemble that of the picture taken, i.e. the mesh in Figure 7b, seems to have the biggest difference among the best scoring tests (+31.87%).

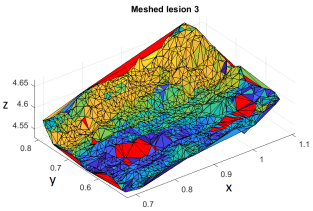
(a) Stereo image.



(b) Mesh.

Fig. 4: Results of test with multiple red lesions on light skin tone, stereo pair 2.

(a) Stereo image.



(b) Mesh.

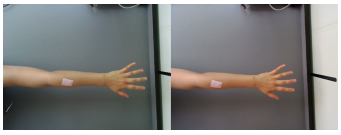
Fig. 5: Results of test with red lesion on dark skin tone, stereo pair 3.

In Figure 4, the results of the second test of the multiple red lesions on the light skin can be seen. It can be seen that the lesions have been detected, but the mesh is not as rectangular

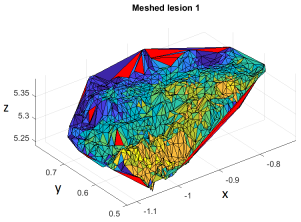
TABLE I: Results of the calculated areas, detected erythema intensity and calculated PASI score.

Lesion colour	Skin tone	Stereo pair	Used colour for detection	Calculated area (cm^2)	Actual area (cm^2)	Difference (%)	Average difference (%)	Area arm point cloud (cm^2)	Detected erythema intensity	PASI score
Red	Light	1	Cr	12.65	12	+ 5.43	+ 29.70	411.40	4	0
		2	Cr	11.94		- 0.47		483.80	4	0
		3	Cr	18.50		+ 54.13		411.74	4	0.8
		4	Cr	19.27		+ 60.59		479.12	4	0.8
		1	Cr	18.22	18	+ 1.23	+ 16.59	352.12	4	0.8
		2	Cr	18.19		+ 1.05		430.52	4	0.8
		3	Cr	18.39		+ 2.19		393.34	4	0.8
		4	Cr	29.14		+ 61.89		338.40	4	0.8
	Dark	1	Cr	12.79	12	+ 6.58	+ 4.18	488.91	4	0
		2	Cr	12.91		+ 7.65		381.59	4	0
		3	Cr	11.48		- 4.29		364.57	4	0
		4	Cr	12.81		+ 6.78		449.22	4	0
Pink	Light	1	Cb	11.03	12	- 8.06	- 5.06	453.25	2	0
		2	Cb	3.67		- 69.36		389.48	2	0
		3	Cb	13.85		+ 15.47		370.25	2	0
		4	Cb	17.00		+ 41.72		280.84	2	0.4
	Dark	1	Y	43.30	12	+ 217.03	+ 141.86	507.43	4	0.8
		2	Y	45.28		+ 277.30		585.17	4	0.8
		3	Y	16.95		+ 41.25		468.46	4	0
		4	Y	15.82		+ 31.87		410.87	4	0

as the stickers actually are. In Figure 5, the results of the third test of the red lesion on dark skin tone can be seen. The shape of this mesh resembles that of the used sticker. However, there are a few holes, which have been detected as excess area.



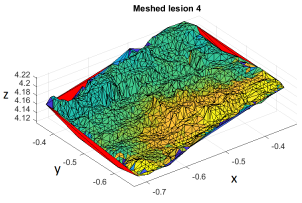
(a) Stereo image.



(b) Mesh.

Fig. 6: Results of test with pink lesion on light skin tone, stereo pair 1.

(a) Stereo image.



(b) Mesh.

Fig. 7: Results of test with pink lesion on dark skin tone, stereo pair 4.

In Figure 6, the results of the first test of the pink lesion on light skin tone can be seen. The shape does not completely resemble that of the sticker, and areas in the middle of the mesh were detected as excess areas. In Figure 7, the results of the fourth test of the pink lesion on dark skin tone can be seen. There were no excess areas detected in the middle of the mesh, only at the outside. The shape resembles that of the sticker.

The MATLAB scripts used to obtain the results can be found in Appendix B.

V. Discussion

As observed in the previous section, the calculated areas differ from the actual area. In general, it seems that the calculated area of the meshes is too large. This is because the obtained point clouds are too textured, resulting in a mesh with many peaks. Since the used stickers do not have any texture, it can be said that the point cloud and thus the mesh are inaccurate. This texture could perhaps be caused by quality of the disparity maps. These maps are not ideal and still have some pixels of different values in the area of interest. This would cause some points to be at a different depth than the others, explaining the peaks as seen in the mesh figures. Generally, disparity maps look worse for non-textured areas, which could also explain the noise in the background area. This noise is also present since the disparity range is not well defined for that area.

Detecting the lesion correctly does not always work. This could be caused by, for example, a wrongly taken picture, meaning part of the lesion is not visible in the pictures. Another possibility is that the sticker reflects light, which in the picture is seen as a lighter colour, making it more difficult to detect. This also causes holes in the point clouds. These holes could also be caused by occlusions in the disparity maps, which after filtering the point cloud results in even bigger occlusions, thus bigger holes in the lesion. Skin could also reflect light, making it particularly hard for light lesions to be detected, as the reflecting skin could be potentially seen as a lesion (e.g. Appendix A-E).

The detection of excess area does not work completely as intended. In many meshes the holes in the middle are coloured

red when they should not be. Detecting these triangles should be further looked into. Another option is to look into other triangulation methods, which do not compute the convex hull, but just the correct area.

There cannot be said much about the accuracy of the calculated arm areas, but the results are likely to be inaccurate due to the point clouds being too textured. In every picture, the arm is not in the exact same position, which could explain why the values differ for every test. Using these areas to calculate the PASI scores by multiplying them by four defeats the whole purpose of why stereo vision is used, since doing so makes the end result imprecise. The calculated PASI scores have not been validated by comparing the results to existing methods. Furthermore, the predetermined areas are too small to calculate a PASI score, as for 12 cm^2 this would be equal to 0 anyway. It would be interesting to evaluate the method by looking at larger lesions. For the overall picture these scores have been added to the results table and could be used as reference for future work, to see how much the accuracy of the score improves for a more complete model.

As for the erythema intensity detection, comparing the reference images to the taken images, the results for the red lesions seem to be fair. For the pink lesion on light skin this detection also seems to work fine. However, for the pink lesion on the dark skin the detected erythema intensity is obviously incorrect. This is because for detecting this lesion the Y colours are examined for the point cloud as well as the reference images for the erythema. An image in Y colours is similar to a gray scale image, where the lightest colours are the brightest. Looking at the reference pictures in Figure 2, the skin of the person in Figure 2d has the lightest colour of all four images, and thus has the highest Y value. Comparing this to the lesion as bright as the one used for these tests, it is clear why the wrong intensity value was detected. Furthermore, it is not clear if this method will work for the other intensities, as this has not been tested.

From the tests, it seems that the method works the best for the red lesion on dark skin. However, only four pictures for each test have been processed, which is not enough to conclude on which colours the method works best.

VI. Recommendations

To improve the presented method, it should be investigated why the point clouds are textured more than they should be. Different disparity methods could be tested, and perhaps even different cameras. Cameras with a higher resolution could help with obtaining better and more detailed point clouds. Additionally, the method should be tested for more lesion colours, as in this paper only two colours have been tested. It would be even better if the method could be tested on actual lesions, not just stickers. Testing the method on the other body parts should be done as well. The erythema intensity detection could be improved by using deep learning, which would be more reliable than the current method presented in this paper. Furthermore, to make this method into an actual PASI determination method, the induration and desquamation intensity should be detected as well. For calculating the induration the point clouds have to be improved, as the depth of the lesion cannot be accurately determined if the point cloud has random peaks. For determining the scaling intensity, deep learning could be used here as well. Considering the area calculation, the method is not complete yet. Ideally, complete 360 degree point clouds of the human body parts are used. This could be realised by using multi-view stereo reconstruction. In addition, the point clouds should be examined for accuracy by comparing them to point clouds obtained from state-of-the-art camera setups.

VII. Conclusion

By using a stereo camera, a 3D model of a human arm can be made and its area can be calculated. Different coloured lesions present on the skin can be detected and examined for area. Using only a stereo camera and a computer makes the method portable and easy to use. Economically, this method would be significantly more affordable than existing reconstruction systems. However, many improvements should be done before the method can actually be used to accurately calculate PASI scores.

References

[1] Hamilton Dr Amanda Oakley, Dermatologist. Pasi score, 2009. https://www.dermnetnz.org/topics/pasi-score/, Last accessed on 2019-06-01.

[2] Fuchs T. Enk A. et al Fink, C. Design of an algorithm for automated, computer-guided pasi measurements by digital image analysis. *Journal of Medical Systems*, 42:248, 2018.

[3] Jonathan Wells Philip Treleaven. 3d body scanning and healthcare applications. *Computer*, pages 28–34, 2007.

[4] Heiko Hirschmüller. Stereo processing by semi-global matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007.

[5] Heiko Hirschmüller. Semi-global matching - motivation, developments and applications. *Photogrammetric Week*, 2011.

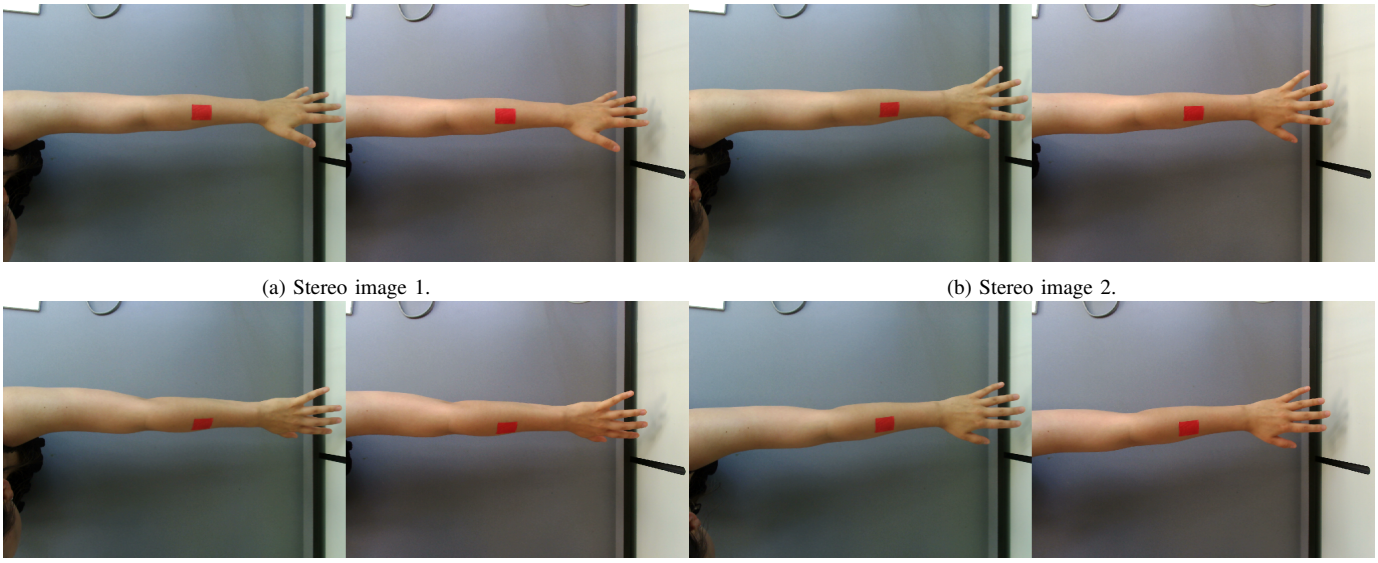
[6] Herbert Edelsbrunner. Triangulations and meshes in computational geometry. *Acta Numerica*, pages 1–81, 2000.

[7] Makarand Tapaswi. Why the rgb to ycbcr, 2009. https://makarandtapaswi.wordpress.com/2009/07/20/why-the-rgb-to-ycbcr/, Last accessed on 2019-06-28.

APPENDIX A
ADDITIONAL RESULTS

In this appendix, the point clouds of all tests will be shown, together with their corresponding stereo pair.

A. *Red lesion on light skin shade*



(a) Stereo image 1.

(b) Stereo image 2.

(c) Stereo image 3.

(d) Stereo image 4.

Fig. 8: Stereo images.

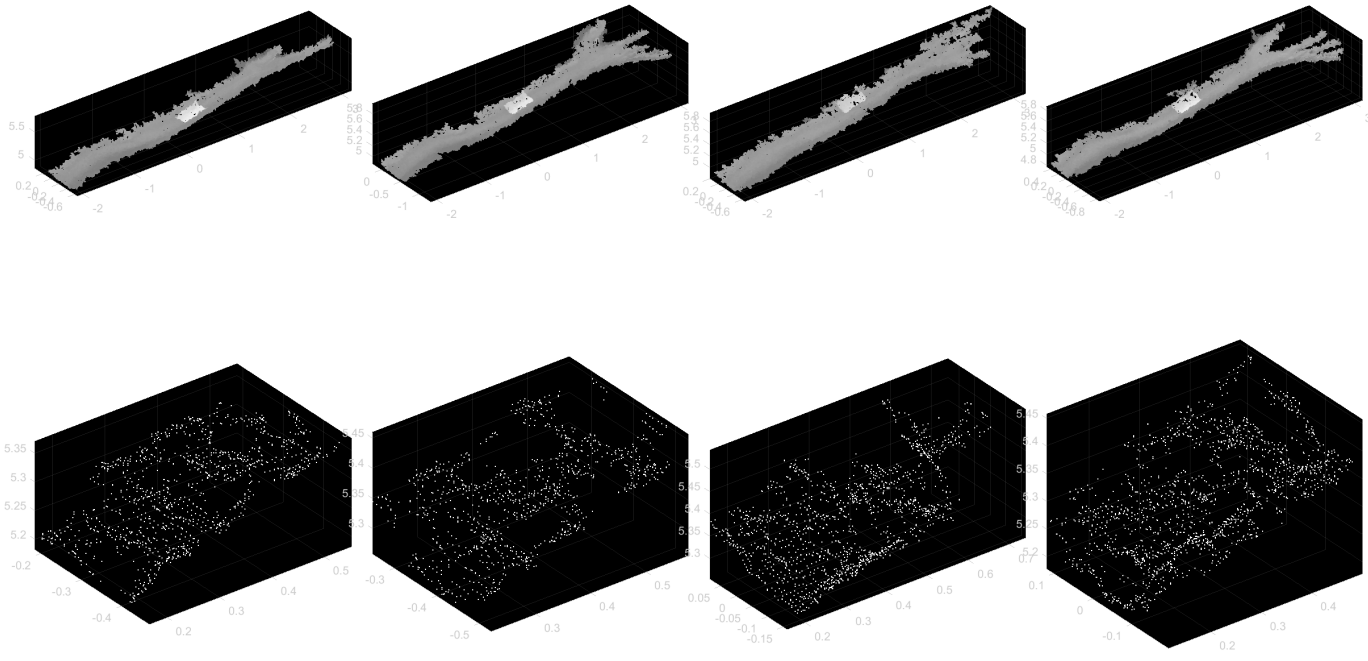
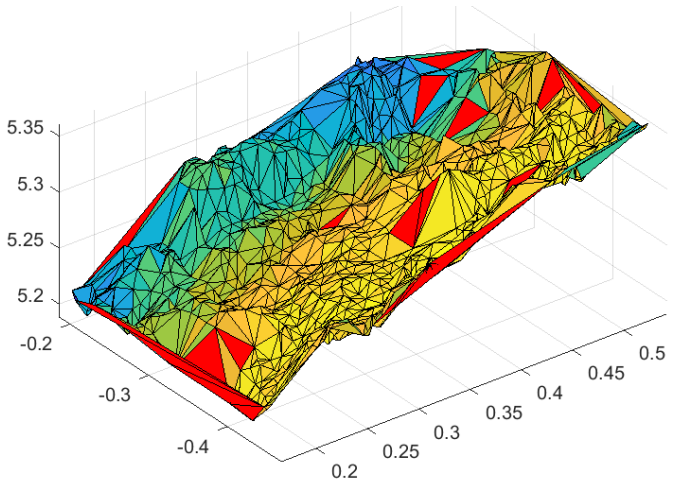
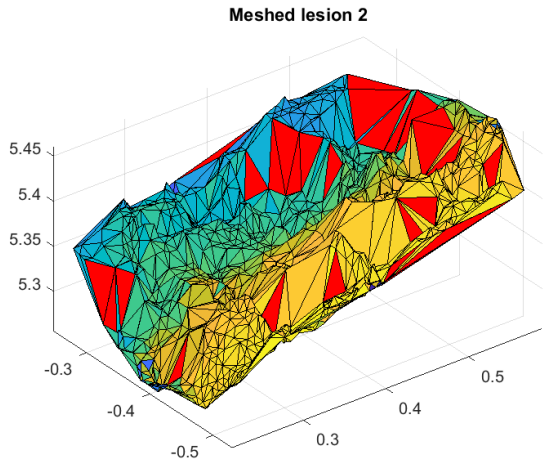


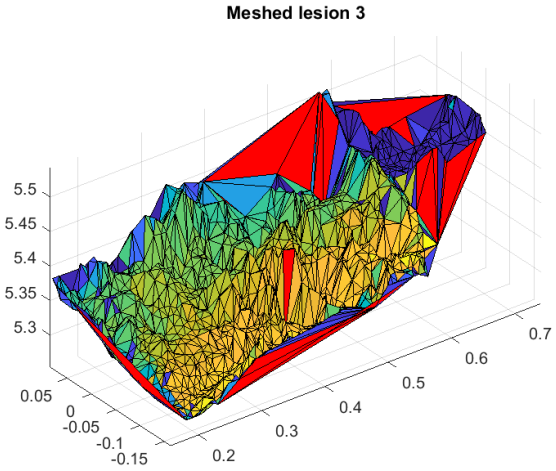
Fig. 9: Point clouds of arm and lesion.



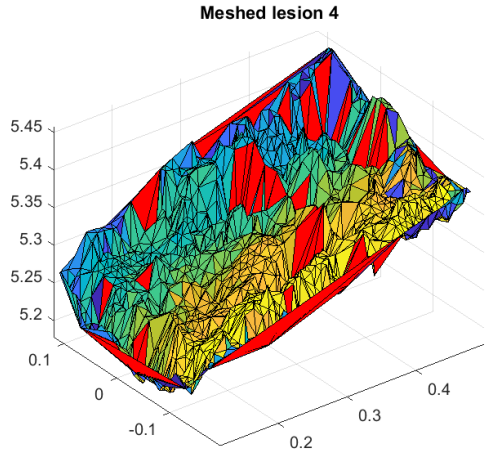
(a) Mesh 1.



(b) Mesh 2.



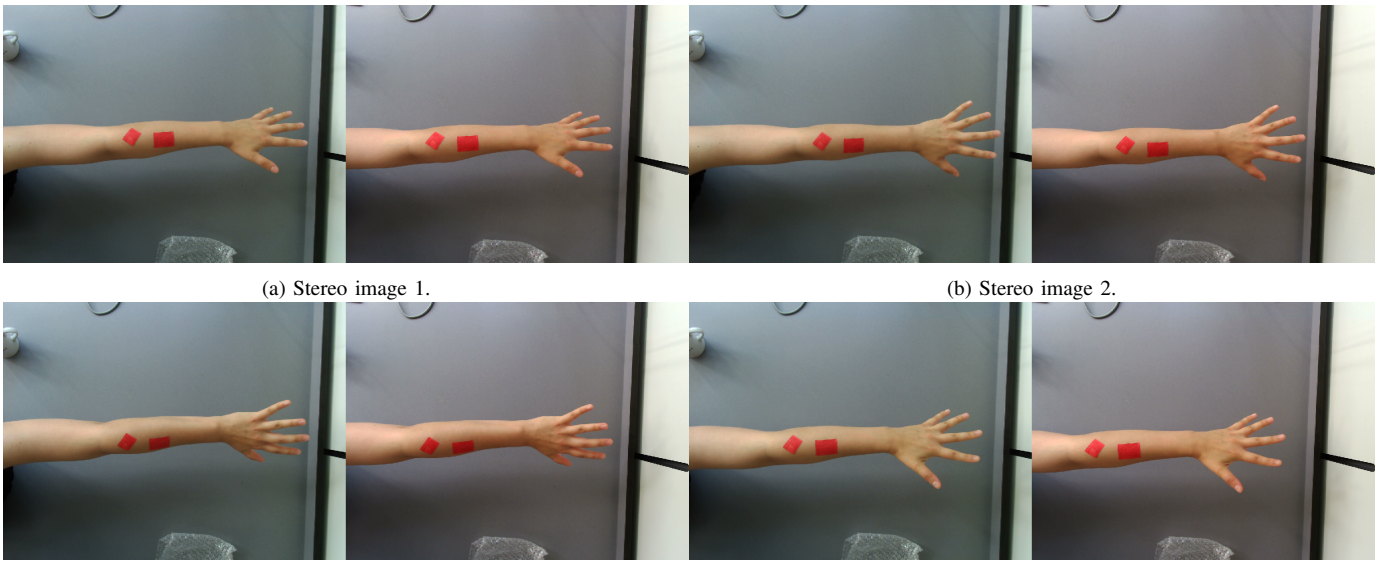
(c) Mesh 3.



(d) Mesh 4.

Fig. 10: Meshes of the detected lesions.

B. Multiple red lesions on light skin shade



(a) Stereo image 1. (b) Stereo image 2.

(c) Stereo image 3. (d) Stereo image 4.

Fig. 11: Stereo images.

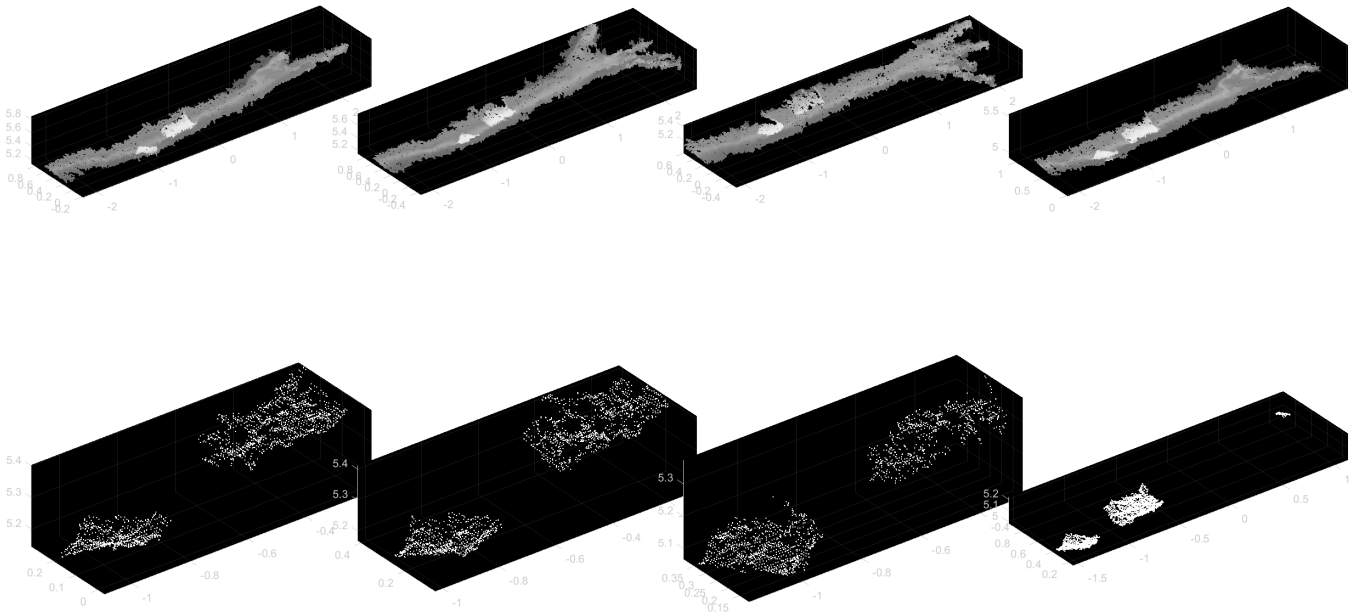
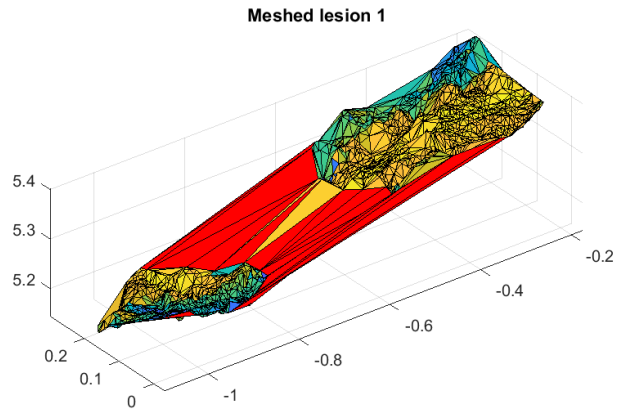
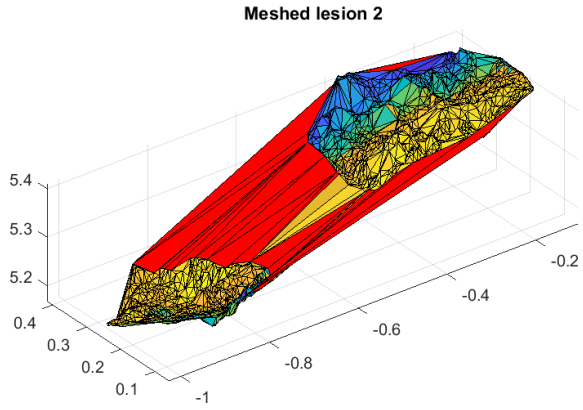


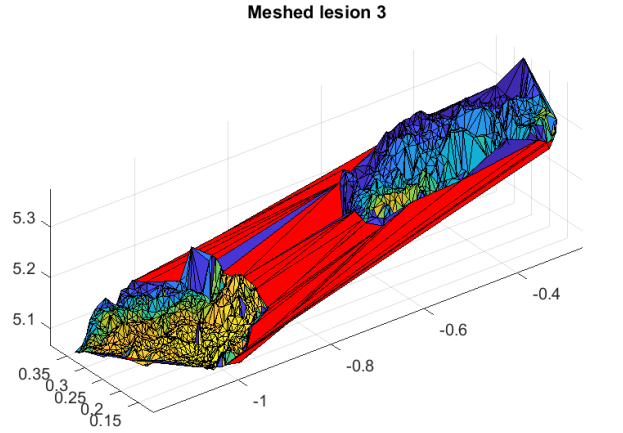
Fig. 12: Point clouds of arm and lesion.



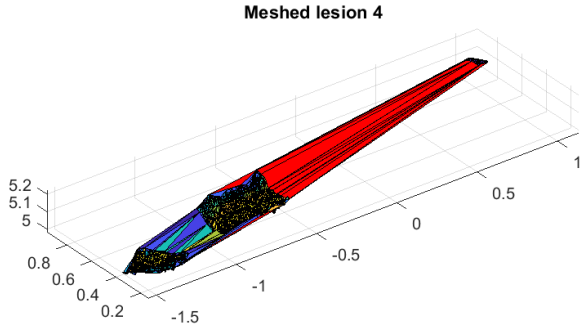
(a) Mesh 1.



(b) Mesh 2.



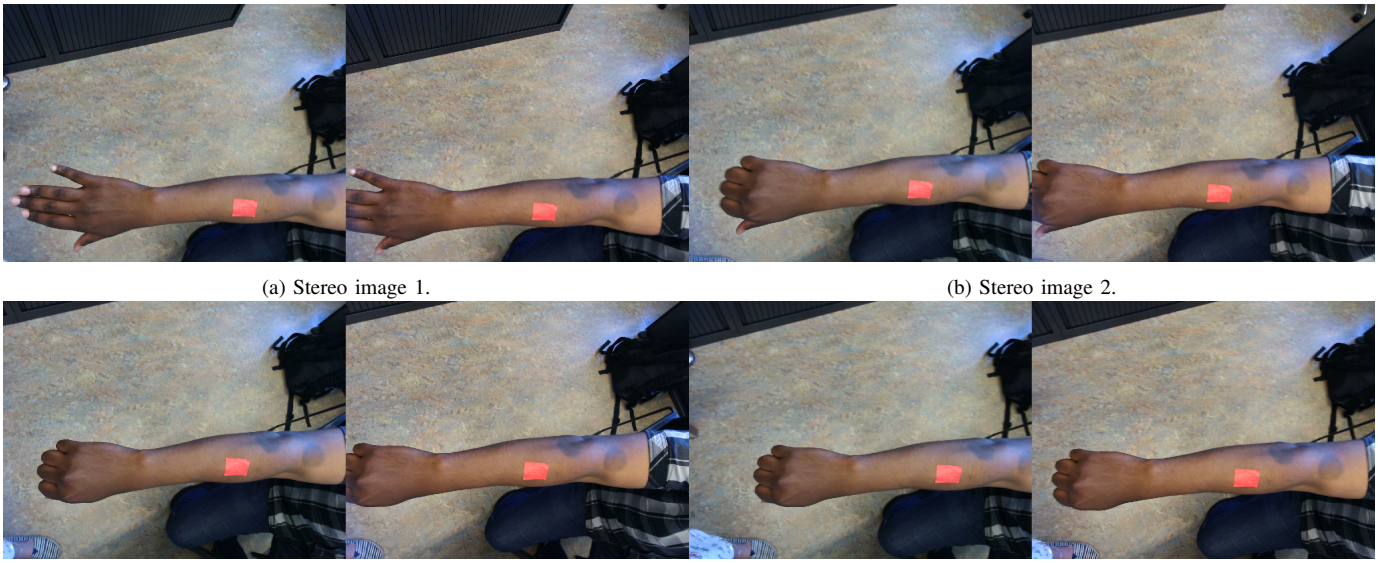
(c) Mesh 3.



(d) Mesh 4.

Fig. 13: Meshes of the detected lesions.

C. Red lesion on dark skin shade



(a) Stereo image 1. (b) Stereo image 2.

(c) Stereo image 3. (d) Stereo image 4.

Fig. 14: Stereo images.

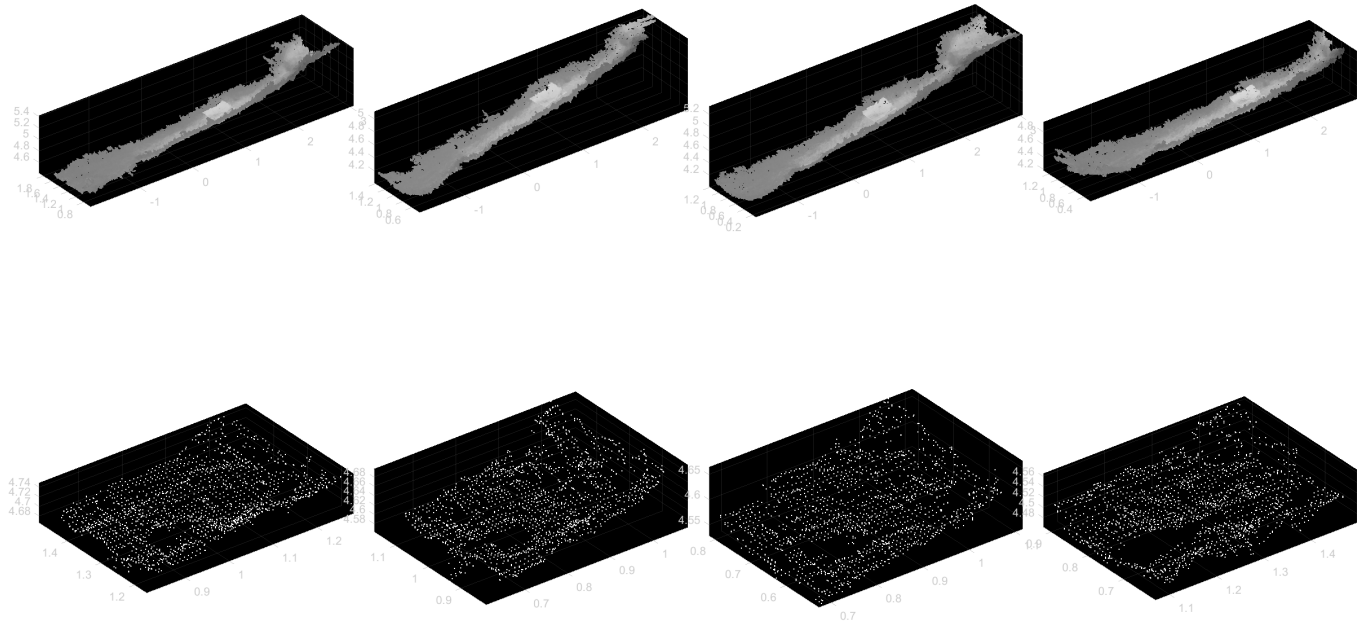
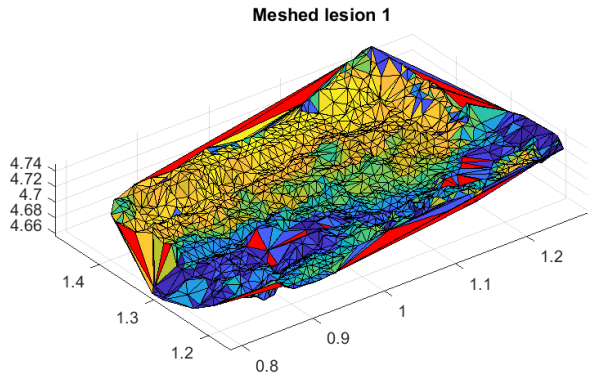
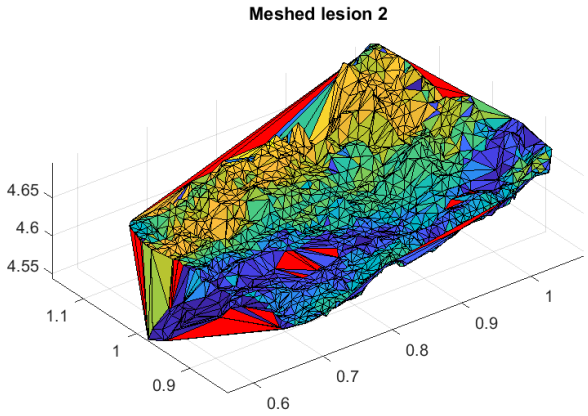


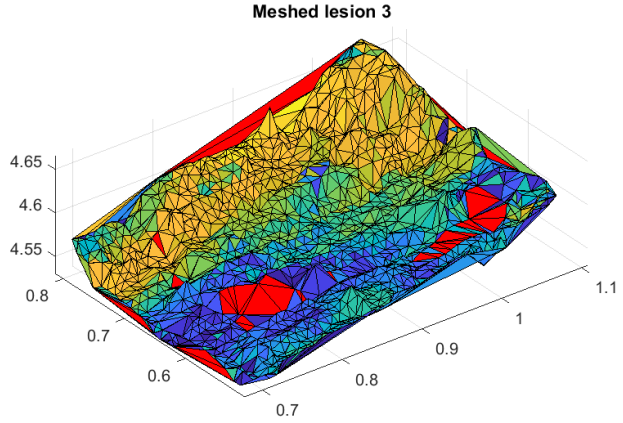
Fig. 15: Point clouds of arm and lesion.



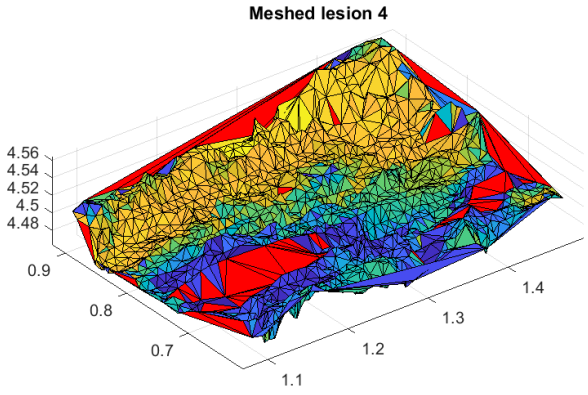
(a) Mesh 1.



(b) Mesh 2.



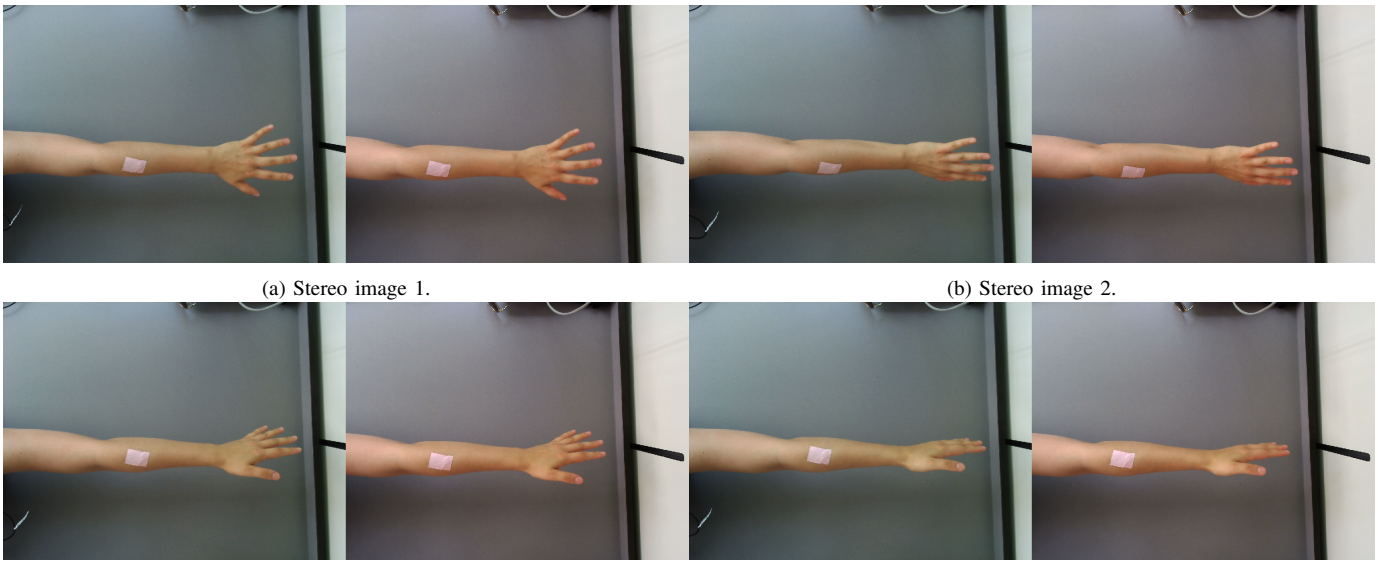
(c) Mesh 3.



(d) Mesh 4.

Fig. 16: Meshes of detected lesions.

D. Pink lesion on light skin shade



(a) Stereo image 1. (b) Stereo image 2.

(c) Stereo image 3. (d) Stereo image 4.

Fig. 17: Stereo images.

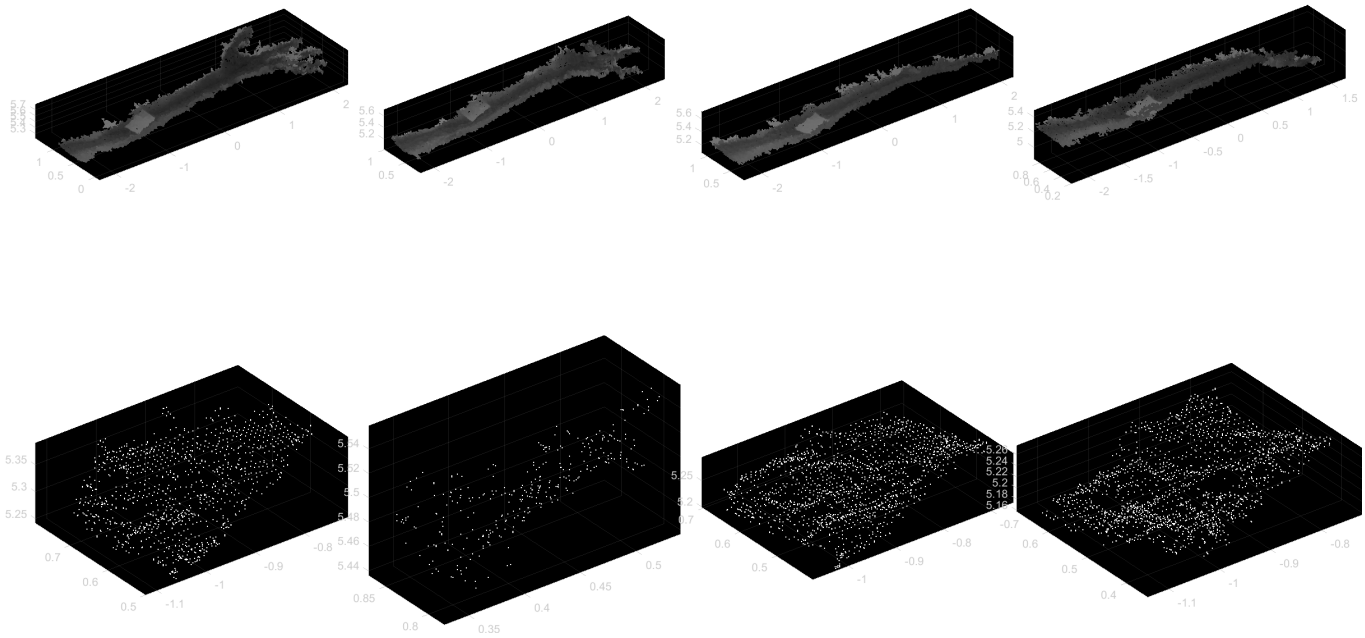
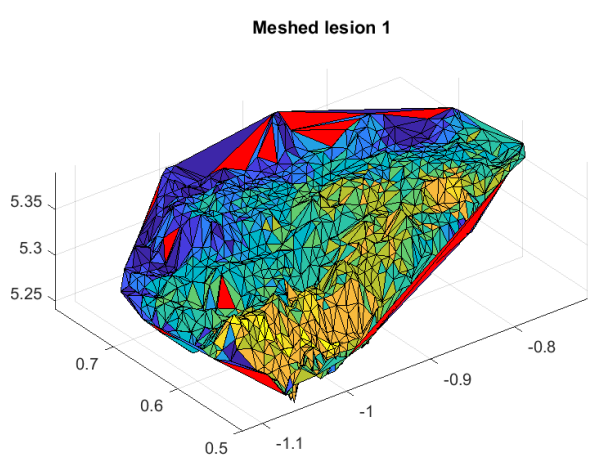
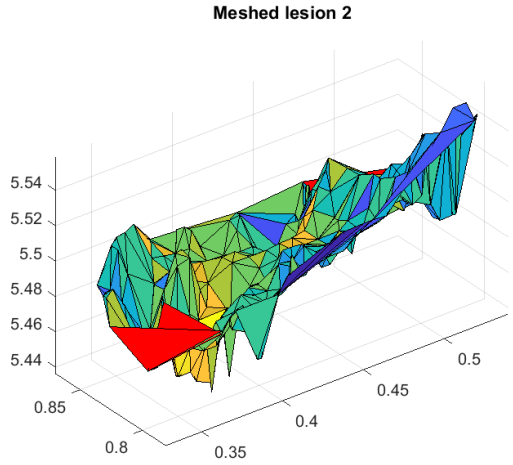


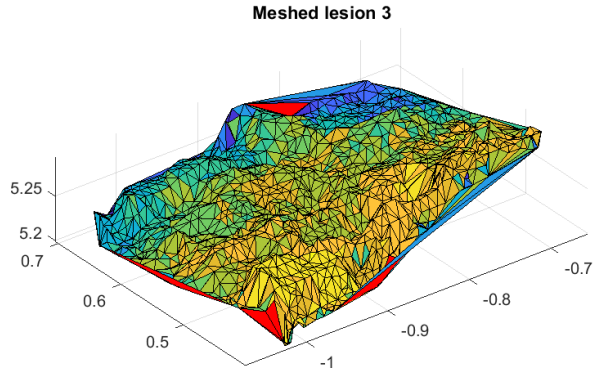
Fig. 18: Point clouds of arm and lesion.



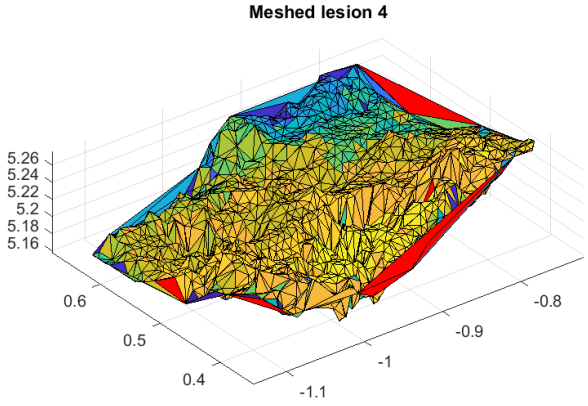
(a) Mesh 1.



(b) Mesh 2.



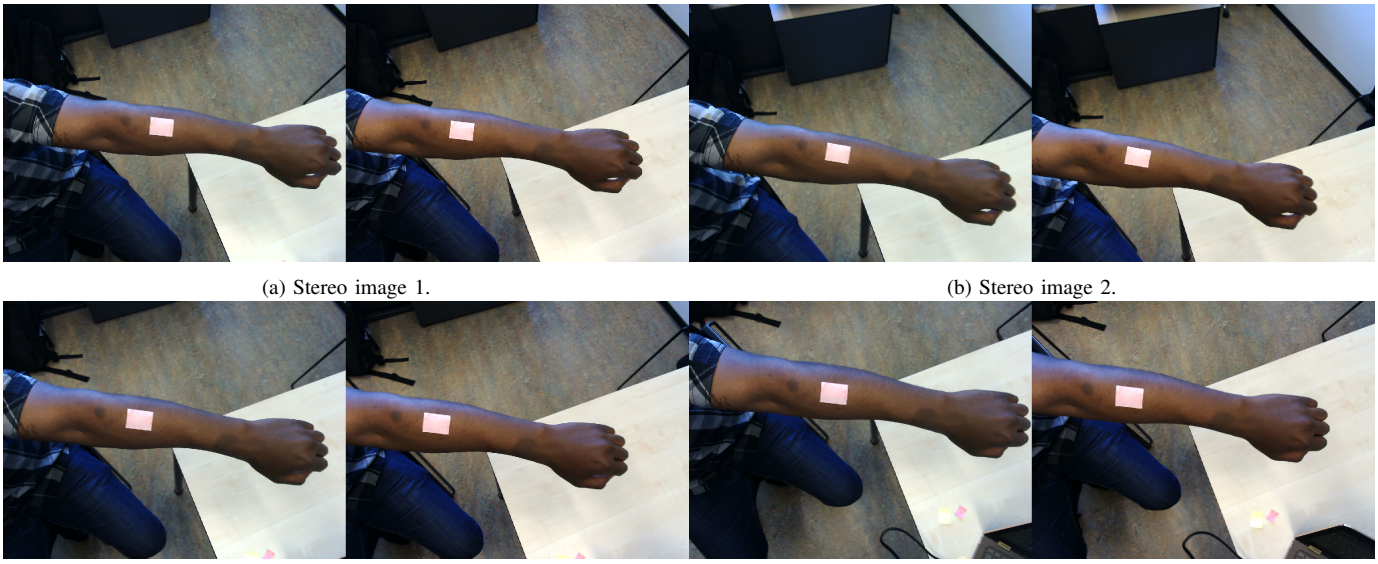
(c) Mesh 3.



(d) Mesh 4.

Fig. 19: Meshes of the detected lesions.

E. Pink lesion on dark skin shade



(a) Stereo image 1. (b) Stereo image 2.

(c) Stereo image 3. (d) Stereo image 4.

Fig. 20: Stereo images.

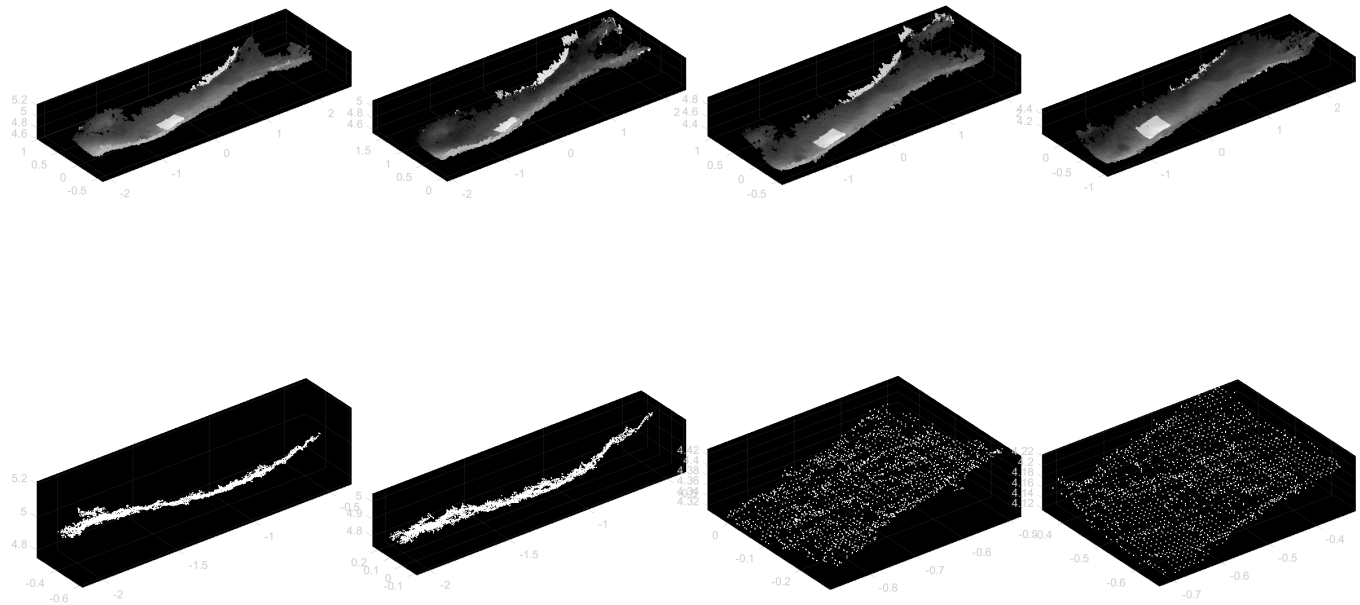
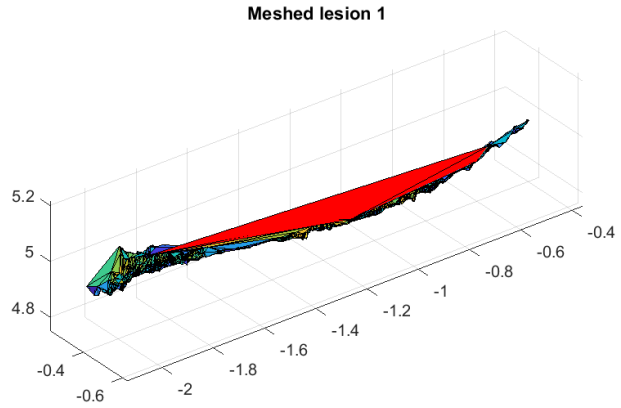
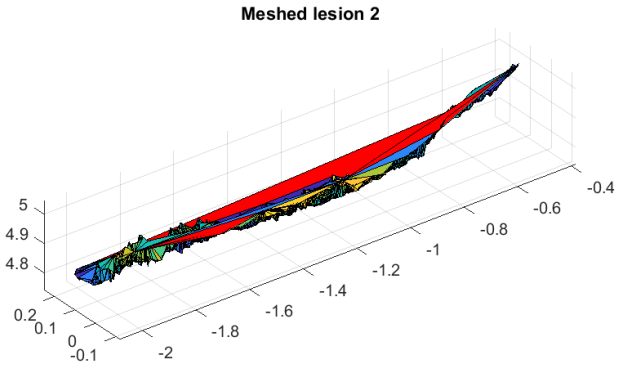


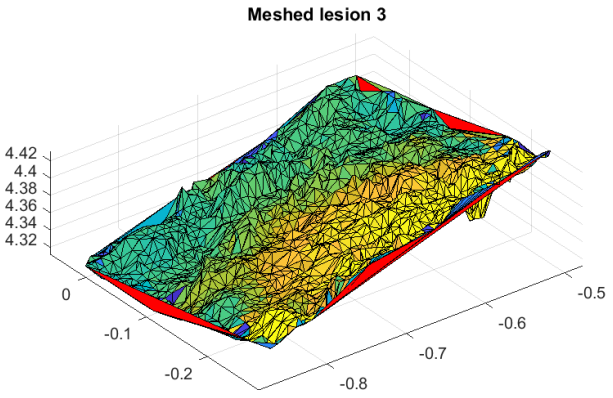
Fig. 21: Point clouds of the arm and lesion.



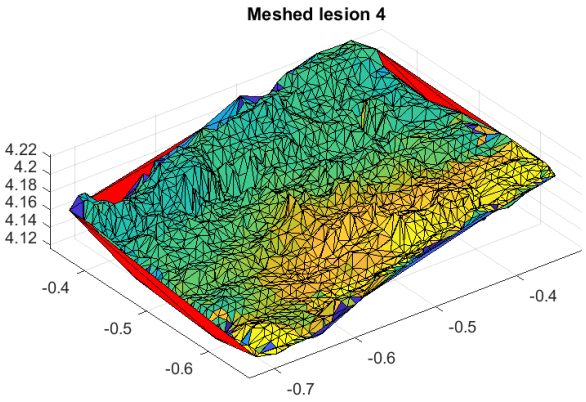
(a) Mesh 1.



(b) Mesh 2.



(c) Mesh 3.



(d) Mesh 4.

Fig. 22: Meshes of the detected point clouds.

APPENDIX B
MATLAB SCRIPTS

In this appendix, the MATLAB scripts are presented. A description is added at the top of each function, clarifying what it does.

A. *sr_functions.m*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This code detects lesions on body parts, using stereo images.
% The pictures may be taken beforehand

% need to have the following functions in the path:
% read_camera: take pictures and save them,
% disparityRange: determine the disparity range of the stereo pair,
% largestCluster: select the largest cluster of a point cloud,
% processPtCloud: compute the denoised point cloud from stereo images,
% surfArea3: compute the area of the mesh,
% goodArea: compute the correct area of the mesh,
% erythemaScore: determine the severity of the erythema (redness),
% intensity: calculate intensity of the area,
% pasi: determine pasi score,

% IMPORTANT NOTE: MAKE SURE THE STEREO CAMERA IS CALIBRATED AND THAT THE
% STEREOPARAMETERS are saved as 'stereoParams.mat' in the path

clear all;
close all;
imtool close all;
set(gcf,'color','w');
fontSize = 15;
tic % starts counting processing time

q.newpic = questdlg('Would you like to take new pictures?',...
    'Yes','No');

% Handle response
switch q.newpic
    case 'Yes'    % take new pictures !! MAKE SURE OLD PICTURES ARE SAVED!!
        q.newp = 1;
    case 'No'     % use old pictures; there should be pictures in the path
        q.newp = 0;
end

if q.newp == 1
[LEFT,RIGHT,n] = read_camera;
elseif q.newp == 0
n = 1;
% n is the number of stereo pairs to be processed, it's automatically
% determined if new pictures are taken
end

%import the stereoparameters into the workspace
load('stereoParams.mat');  % make sure the .mat file is in the right folder

% read images from the workspace
for i = 1 : n
stereoIm.LEFT.rgb{i} = imread(sprintf('TESTL_%s.png', num2str(i)));
stereoIm.RIGHT.rgb{i} = imread(sprintf('TESTR_%s.png', num2str(i)));
figure; imshow([stereoIm.LEFT.rgb{i},stereoIm.RIGHT.rgb{i}]);
end

%% Determine disparity ranges
disp("disparity");
% disparity will be a cell containing all determined disparity ranges
disparity = disparityRange(n,stereoIm.LEFT.rgb,stereoIm.RIGHT.rgb,...
    stereoParams);
%% point cloud processing
disp("pointcloud");
% point.all will be a cell containing all denoised point clouds
point.all = processPtCloud(n,stereoIm.LEFT.rgb,stereoIm.RIGHT.rgb,...
    stereoParams,disparity);
%% delete invalid point clouds
 % makes figure full-screen
```

```
figure('units','normalized','outerposition',[0 0 1 1]);
for l = 1 : n
subplot(ceil(n/2),ceil(n/2),l)
pcshow(point.all{l});
t = sprintf('Point cloud %d', l);
title(t);
end

% if there's a faulty point cloud, this is the time to remove it
prompt = sprintf(['Delete which point clouds? (type one number,' ...
    'space separated numbers, or leave the space empty)']);
q.delnum = inputdlg(prompt);
delete = str2num(q.delnum{1});
selected = zeros(1,n);        % if n = 5, (0,0,0,0,0)

for m = 1:n
    selected(m) = m;          % (1,2,3,4,5)
end

% say we want to remove 2,3, selected = (1,4,5)
selected(delete) = [];

for m = 1:size(selected,2)
% pointSelect is only the 1st 4th and 5th point clouds
pointSelect{m} = point.all{selected(m)};
end

for l = 1 : m
    subplot(ceil(m),ceil(m/2),l)
    pcshow(pointSelect{l});
    t = sprintf('Point cloud %d.', selected(l));
    title(t);
end

%% process point clouds
for h = 1:size(pointSelect,2)
    % smooth point cloud arm
    pointArm = pointSelect{h};
    pointArmSmooth{h} = medfilt3(pointArm.Location,[5,1,7]);
    figure; pcshow(pointArmSmooth{h});
    tl = sprintf('Smoothed point cloud %d', h);
    title(tl);

    % Depending on the lesion colour, the point cloud colour will be chosen
    % to be further processed. Choose between Y Cb and Cr, look for which
    % colour the lesion is the brightest and contrasts the arm the most
    figure;
    % point cloud in Y
    colory = uint8(240*mat2gray(rgb2ycbcr(im2double(pointArm.Color))));
    pointArmYcbcr3{h} = pointCloud(pointArm.Location,'Color',...
        [colory(:,1),colory(:,1),colory(:,1)]);
    subplot(2,2,1); pcshow(pointArmYcbcr3{h}); axis equal
    title('1');

    % point cloud in Cb
    colory = uint8(240*mat2gray(rgb2ycbcr(im2double(pointArm.Color))));
    pointArmYcbcr2{h} = pointCloud(pointArm.Location,'Color', ...
        [colory(:,2),colory(:,2),colory(:,2)]);
    subplot(2,2,2); pcshow(pointArmYcbcr2{h}); axis equal
    title('2');

    % point cloud in Cr
    colory = uint8(240*mat2gray(rgb2ycbcr(im2double(pointArm.Color))));
    pointArmYcbcr1{h} = pointCloud(pointArm.Location,'Color', ...
        [colory(:,3),colory(:,3),colory(:,3)]);
    subplot(2,2,[3,4]); pcshow(pointArmYcbcr1{h}); axis equal
    title('3');

    prompt = ['In which pointcloud is the lesion the brightest',...
        'and has the most contrast to the skin (1, 2 or 3)?'];
    q.l = inputdlg(prompt);
    q.light = str2num(q.l{1});

    if q.light == 1 %Y; works best for light lesions
        % look at YCBCR and construct lesion pointcloud
        pointArmYcbcr{h} = pointArmYcbcr3{h};
        range = 2;
```

```
        LES = pointArmYcbcr{h}.Color ≥ max(max(colory(:,3))) - range;
    elseif q.light == 2 %CB
        % look at YCBCR and construct lesion pointcloud
        pointArmYcbcr{h} = pointArmYcbcr2{h};
        range = 25;
        LES = pointArmYcbcr{h}.Color ≥ max(max(colory(:,2))) - range;
    elseif q.light == 3 %CR; works best for red lesions
        % look at YCBCR and construct lesion pointcloud
        pointArmYcbcr{h} = pointArmYcbcr1{h};
        range = 60;     % 30 for dark skin, 60 for lighter
        LES = pointArmYcbcr{h}.Color ≥ max(max(colory(:,3))) - range;
    end

    [ROWS,COLS] = find(LES == 1); % obtain the rows with the right colour
    pointLesionRgb = select(pointArmYcbcr{h},ROWS); % make the point cloud
    figure; pcshow(pointLesionRgb); axis equal
    title('Pointcloud of lesion before filtering');

    % denoise pointcloud
    % outlier threshold. A point is considered an outlier if the
    % average distance to numNeighbors > threshold
    pointLesionDenoised{h} = pcdenoise(pointLesionRgb, 'NumNeighbors', ...
        100, 'Threshold', 0.01); % removes outliers from the pointcloud
    figure;pcshow(pointLesionDenoised{h}); axis equal;
    title('Denoised pointcloud lesion')

    % if there are clusters of which the lesion is the largest, this can be
    % filtered out
    q.cluster = questdlg('Use cluster detection?',...
                         'Yes','No');
    % Handle response
    switch q.cluster
        case 'Yes'          % will detect largest cluster
            q.cl = 1;
        case 'No'           % nothing changes
            q.cl = 0;
    end

    if q.cl == 1
        pointLesionCluster = largestCluster(0.02,pointLesionDenoised{h});
        figure;pcshow(pointLesionCluster);
        title('Cluster detected lesion');
    else
        pointLesionCluster = pointLesionDenoised{h};
    end

    % smooth lesion surface by using median filter
    pointLesionSmooth{h} = medfilt3(pointLesionCluster.Location,[3,1,3]);
    figure;pcshow(pointLesionSmooth{h},'r');
    title('lesionsmooth');

    % delaunay triangulation of whole pointcloud
    disp("triangulation arm");
    pointcloudTempA = pointArmSmooth{h};
    % obtain x y and z coordinates of the vertices
    triangulation.armX = double(pointcloudTempA(:,1));
    triangulation.armY = double(pointcloudTempA(:,2));
    triangulation.armZ = double(pointcloudTempA(:,3));

    % delaunay outputs a matrix of m x 3 where m is the number of triangles
    triangulation.arm = delaunay(triangulation.armX,triangulation.armY);
    figure; triplot(triangulation.arm,triangulation.armX,...
        triangulation.armY);

    % surface mesh
    figure; trisurf(triangulation.arm,triangulation.armX,...
        triangulation.armY,triangulation.armZ, colory(:,3)); axis equal;
    xlabel('x');
    ylabel('y');
    zlabel('z');
    caption = sprintf('Surface meshed point cloud');
    title(caption, 'FontSize', fontSize);

    % triangulation of lesion
    disp("triangulation lesion");
    pointcloudTempL = pointLesionSmooth{h};
    % obtain x y and z coordinates of the vertices
```

```
    triangulation.lesionX = double(pointcloudTempL(:,1));
    triangulation.lesionY = double(pointcloudTempL(:,2));
    triangulation.lesionZ = double(pointcloudTempL(:,3));

    % delaunay outputs a matrix of m x 3 where m is the number of triangles
    triangulation.lesion = delaunay(triangulation.lesionX,...
        triangulation.lesionY);
    figure; triplot(triangulation.lesion,triangulation.lesionX,...
        triangulation.lesionY);

    % surface mesh
    figure; trisurf(triangulation.lesion,triangulation.lesionX,...
        triangulation.lesionY,triangulation.lesionZ, ...
        pointLesionCluster.Color(:,3)); axis equal;
    set(gcf, 'Renderer', 'zbuffer');
    xlabel('x');
    ylabel('y');
    zlabel('z');
    caption = sprintf('Surface meshed point cloud');
    title(caption, 'FontSize', fontSize);


    % calculate surface area
    disp("area arm");
    % obtain the area, the colored triangles and their vertice coordinates
    [surface.arm] = surfArea3(triangulation.arm, triangulation.armX,...
        triangulation.armY, triangulation.armZ);
    area.arm{h} = sum(surface.arm);  % total area, including wrong triangles
    axis equal

    %%% NOTE: AREA IS IN SQUARE DECIMETERS %%%

    % remove the excess triangles
    disp("correctArea arm");

    figure;trisurf(triangulation.arm,triangulation.armX,...
        triangulation.armY,triangulation.armZ,...
        pointArmYcbcr{h}.Color(:,3)); hold on;
    % goodArea selects which triangles do not belong
    surface.armCorrect = goodArea(triangulation.arm,triangulation.armX,...
        triangulation.armY,triangulation.armZ,surface.arm,...
        pointArmYcbcr{h},17,2);

    % total area excluding the large excess triangles
    area.armCorrectDM{h} = sum(surface.armCorrect);
    area.armCorrectCM{h} = area.armCorrectDM{h} * 100; % in cm^2

    % area lesion
    disp("area lesion");
    % obtain the area, the colored triangles and their vertice coordinates
    [surface.lesion] = surfArea3(triangulation.lesion, ...
        triangulation.lesionX, triangulation.lesionY, ...
        triangulation.lesionZ);
    area.lesion{h} = sum(surface.lesion); % total area of triangulation
    axis equal

    disp("correctArea lesion");
    figure;trisurf(triangulation.lesion,triangulation.lesionX,...
        triangulation.lesionY,triangulation.lesionZ,...
        pointLesionCluster.Color(:,3)); hold on;
    title(sprintf('Meshed lesion %d',h));
    surface.lesionCorrect = goodArea(triangulation.lesion,...
        triangulation.lesionX,triangulation.lesionY,...
        triangulation.lesionZ,surface.lesion,...
        pointLesionCluster,3,5); axis equal;
    % for multiple lesions, change 3,5 to 4,3

    % compute total area excluding the large excess triangles
    area.lesionCorrectDM{h} = sum(surface.lesionCorrect);
    area.lesionCorrectCM{h} = area.lesionCorrectDM{h} * 100; % in cm
end

%% process areas
% plot the point clouds of the arm and the lesion in one figure
figure('units','normalized','outerposition',[0 0 1 1]);
for i = 1:h
    if h == 1   % for one stereo pair, plot the clouds next to eachother
```

```
        subplot(1,2,1)
        pcshow(pointArmYcbcr{i});
        title('Point cloud of arm in Cr.')
        subplot(1,2,2);
        pcshow(pointLesionSmooth{i},'w');
        title('Point cloud of detected lesion.')
    else
        subplot(ceil(h/2),ceil(h),i)
        pcshow(pointArmYcbcr{i});
        subplot(ceil(h/2),ceil(h), h + i)
        pcshow(pointLesionSmooth{i},'w');
    end
end

% calcuate the average of the arm and lesion areas
sumA = 0;
sumL = 0;
for p = 1:h
    sumA = sumA + area.armCorrectDM{h};
    sumL = sumL + area.lesionCorrectDM{h};
end
area.armAverage = sumA/h;
area.lesionAverage = sumL/h;
% assume that the area of the arm as calculated here is about half of the
% total. So for two arms, area_good*4


% how much area the wound is compared to the arm (in percent)
%F = (area.lesionAverage/area.armTotal)*100;

for i = 1:n
% return intensity score (0-4)
area.armTotal = area.armCorrectCM{i}*4;
F = (area.lesionCorrectCM{i}/area.armTotal) * 100;
arm.intensity.area{i} = intensity(F);

% determine erythema severity
arm.intensity.erythema{i} = erythemaScore(pointLesionDenoised{i},q.light);
%pointLesionDenoised is the YCBCR colored cloud of the lesion

% calculate pasi score
PASI.head = [0,0,0,0];   % (area, erythema, induration, desquamation)
PASI.arms = [arm.intensity.area{i},arm.intensity.erythema{i},0,0];
PASI.trunk = [0,0,0,0];
PASI.legs = [0,0,0,0];
score{i} = pasi(PASI.head,PASI.arms,PASI.trunk,PASI.legs);

% show pasi score
PASIscore = sprintf('PASI score %d: %d', i, score{i});
box.pasi = msgbox({PASIscore}, 'PASI score');
disp(sprintf('Lesion area = %d', area.lesionAverage))
end

% calculate difference between areas
prompt = 'What is the actual area (in square cm)?';
q.area = inputdlg(prompt);
for i = 1:n
    difference{i} = str2num(q.area{1})-area.lesionCorrectCM{i};
    diffp = round(((abs(difference{i}))/str2num(q.area{1})) * 100,2);
    if difference{i} < 0 % the calculated area is bigger
        text.diff = sprintf(['The calculated area %d is %f percent',...
            'larger than the actual area'], i, diffp);
    else % calculated area is smaller
        text.diff = sprintf(['The calculated area %d is %f percent',...
            'smaller than the actual area'], i, diffp);
    end
    text.qarea = sprintf('The actual area is %f square cm',...
        str2num(q.area{1}));
    text.carea = sprintf('The calculated area is %f square cm',...
        area.lesionCorrectCM{i});
    box.area = msgbox({text.qarea, text.carea, text.diff});
end
totalTime = toc
```

B. *read_camera.m*

```
function [LEFT,RIGHT,n] = read_camera
% read_camera determines the area of a meshed point cloud
% area = goodArea(triangulation,X,Y,Z,A,pointcloud,length,factor))
% - area is the area of the meshed point cloud
% - triangulation is the triangulated point cloud
% - X, Y, Z are the coordinates of the vertices of the triangles
% - A is the area of all triangles summed up
% - pointcloud is the pointcloud of which the triangulation is made
% - length is an integer; the maximum line length is Xlim/length
% - factor is an integer, indicating how much larger a triangle may be
% wrt the mean area
% !!! BEFORE RUNNING THIS CODE MAKE SURE EXISTING IMAGES WON'T BE
% OVERWRITTEN!!!!
clear all
close all

camList = webcamlist;           % shows the list of available cameras
cam = webcam(2);                % make sure to select the stereo camera!
preview(cam);                   % gives a real-time view of the camera
fontSize = 10;

n = 0;
figure;
while 1
    %ask user to take picture
    user = input('Take picture? yes = y, no = n ', 's');
    if user == 'n'
        if n == 0
            f = errordlg('No pictures were taken.');
        %if the user wants to stop taking pictures
        end
        break
    else
        %user wants to take a picture
        n = n + 1;
        picture = snapshot(cam);    % takes a picture
        temp = picture;
        imshow(temp);               % view the pictures before saving them
        % save pictures
        LEFT{n} = picture(:,1:(size(picture,2))/2,:);
        RIGHT{n} = picture(:,((size(picture,2))/2)+1 : size(picture,2),:);
    end
    X = ['You have taken ', num2str(n) ,' picture(s).'];
    disp(X);
end

%show and save the pictures as .png

figure;
for k = 1:n
    if n ≤ 2               % i.e. for only 1 or two pictures
       subplot(n, 1, k);
    else
       subplot(ceil(n/2),ceil(n/2),k);
    end
    %write the pictures to .png files
    imshow([LEFT{k}, RIGHT{k}]);
    imwrite(LEFT{k},sprintf('TESTL_%d.png',k));
    imwrite(RIGHT{k}, sprintf('TESTR_%d.png',k));
    caption = sprintf('Stereo pair #%d of #%d', k, n);
    title(caption, 'FontSize', fontSize);
    hold on
end

end
```

C. *disparityRange.m*

```
function disparity = disparityRange(n,LEFT,RIGHT,stereoParams)
% disparityRange determines the disparity range for stereo pairs, if the
% user wants to make a disparity map of a subject in the foreground.
% disparity = disparityRange(n,LeftIm,RightIm,stereoParameters)
% - disparity a cell containing the disparity ranges for each stereo pair
% - n is the amount of stereo pairs used as input
% - LEFT is a cell containing all left images
% - RIGHT is a cell containing all right images
% - stereoParameters is the parameters obtained from the stereo
% calibration app

for pair = 1: n
    % rectify the images
    [IL{pair},IR{pair}] = rectifyStereoImages(LEFT{pair},RIGHT{pair},...
        stereoParams);
    I = stereoAnaglyph(IL{pair},IR{pair});
    figure;imshow(I);
    msgbox(sprintf(['Please draw a line from a lesion point in the'...
        'left image to the same point in the right image. Then,'...
        'double click the line.']));
    % draw a line on the image
    h = imline;
    position = wait(h);  % then double click on the line

    % obtain 2x2 matrix containing x and y coordinates of the endpoints
    pos = getPosition(h);
    XL = pos(1,1);
    YL = pos(1,2);
    XR = pos(2,1);
    YR = pos(2,2);

    % calculate the lenght of this line
    length = ceil(sqrt((XR-XL)^2+(YR-YL)^2));
    DISPARITY{pair} = [length - 20, length + 68];
end
disparity = DISPARITY;
```

D. *largestCluster.m*

```
function ptCloud = largestCluster(minDistance, pointcloud)
% largestCluster filters the input pointcloud and returns the largest
% cluster
% ptCloud = largestCluster(minDistance, pointcloud)
% - ptCloud = output point cloud
% - minDistance is the minimum Euclidean distance between points from
% two different clusters, specified as a positive scalar
% - pointcloud is the input point cloud

% Segment the point cloud; detect the clusters
[labels,numClusters] = pcsegdist(pointcloud,minDistance);
figure;
pcshow(pointcloud.Location,labels)
colormap(hsv(numClusters))
title('Point Cloud Clusters')

% remove unwanted clusters, keep the largest
% pointcloud.Location contains as much points as there are labels
% connect the indices of the points from labels containing number M, to
% the indices of input pointcloud
M = mode(labels);       % finds the most occuring number in labels;
                        % The Mth cluster is the biggest cluster
F = find(labels == M);  % returns an array of indices containing M
ptCloud = select(pointcloud,F); % select only those points
end
```

E. *processPtCloud.m*

```
function pointcloud = processPtCloud(n,LEFT,RIGHT,stereoParams,dispRange)
% processPtCloud returns the point cloud of a subject in the foreground.
% pointcloud = processPtCloud(n,LEFT,RIGHT,stereoParams,dispRange)
% - pointcloud is the denoised pointcloud
% - n is the amount of stereo pairs used as input
% - LEFT is a cell containing all left images
% - RIGHT is a cell containing all right images
% - stereoParameters is the parameters obtained from the stereo
% calibration app
% - dispRange is the disparity range (e.g. [0,64]), should be a cell

fontSize = 15;

for m = 1 : n
    % rectify the images
    [IL{m},IR{m}] = rectifyStereoImages(LEFT{m},RIGHT{m},stereoParams);
    figure;
    subplot(1,2,1);
    IL{m} = histeq(IL{m});        % preprocess images
    ILB{m} = rgb2gray(IL{m});
    imshow(ILB{m});

    subplot(1,2,2);
    IR{m} = histeq(IR{m});        % preprocess images
    IRB{m} = rgb2gray(IR{m});
    imshow(IRB{m});

    A = stereoAnaglyph(IL{m},IR{m}); % makes a red-cyan overlapping image
    figure; imshow(A);
    caption = sprintf('Anaglyph picture %d', m);
    title(caption, 'Fontsize', fontSize);

    disparityRange = dispRange{m};
    uniqueness = 11;  % low value --> less reliable disparity map

    disparityMap1{m} = disparitySGM(ILB{m},IRB{m},...
        'DisparityRange',disparityRange,'UniquenessThreshold',...
        uniqueness);

    % remove noise in the disparity map
    disparityMap{m} = medfilt2(disparityMap1{m});
    figure; imshow(disparityMap{m},disparityRange);
    caption = sprintf('Disparity Map of picture %d', m);
    title(caption, 'Fontsize', fontSize);
    colormap(gca,jet);
    colorbar;

    %obtain the 3D points from the disparity maps
    points3D = reconstructScene(disparityMap{m},stereoParams);
    pointCl{m} = removeInvalidPoints(pointCloud(points3D ./ 100,...
        'Color',IL{m})); % remove NaN and Inf points
    figure; pcshow(pointCl{m});
    caption = sprintf('Point cloud #%d of #%d', m, n);
    title(caption, 'FontSize', fontSize);

    % preprocess the point cloud
    % downsampling
    ptClDown{m} = pcdownsample(pointCl{m}, 'gridAverage', 0.01);
    figure; pcshow(ptClDown{m});
    caption = sprintf(['Downsampled point cloud #%d of #%d,'...
        'gridsize 0.01'], m, n);
    title(caption, 'FontSize', fontSize);

    % preprocess the point cloud
    % denoising
    % filter by looking at the distance between point clouds
    numNeighbors = 100;  % number of nearest neighbor points
    % outlier threshold. A point is considered an outlier if
    % the avg distance to numNeighbors > threshold
    threshold = 0.01;
    ptCldn{m} = pcdenoise(ptClDown{m}, 'NumNeighbors', numNeighbors,...
        'Threshold', threshold); % removes outliers from the pointcloud
    figure;
```

```
74      pcshow(ptCldn{m});
75
76      % denoising
77      % Segment the point cloud; detect the clusters
78      minDistance = 0.015;          % should be bigger than 'gridAverage'
79      point{m} = largestCluster(minDistance,ptCldn{m}) ;
80
81      figure; pcshow(point{m});
82      caption = sprintf('Downsampled denoised point cloud #%d of #%d', m, n);
83      title(caption, 'FontSize', fontSize);
84
85
86  end
87
88  % write point cloud to .mat
89  pointcloud = point;
90  end
```

F. surfArea3.m

```
function [area] = surfArea3(triangulation, X, Y, Z)
% surfArea3 returns the area of a triangulation; all triangles summed
% area = surfArea3(triangulation, X, Y, Z)
% area = the area of the input triangulation
% - triangulation is the triangulated surface,
% i.e. triangulation = delaunay(X,Y)
% - X,Y,Z are the x y and z coordinates of the vertices

tri = triangulation;
sizetri = size(tri,1);

% total area
A = [];

for i = 1:sizetri
    tri_num = i;   % triangle #
    % each row of tri contains the #th point from the point cloud of which
    % a triangle is made
    % get their coordinates
    tx = X(tri(tri_num,:));  % tx is the x coordinates of the three points
    ty = Y(tri(tri_num,:));
    tz = Z(tri(tri_num,:));

    point1 = [tx(1,:),ty(1,:),tz(1,:)]; %xyz coordinates of point 1
    point2 = [tx(2,:),ty(2,:),tz(2,:)];
    point3 = [tx(3,:),ty(3,:),tz(3,:)];

    % surface area
    %fill3(x,y,z,'r')
    x = tx(:)';                 % transpose tx, ty, tz. x = 1x3 matrix
    y = ty(:)';
    z = tz(:)';
    ons = [1 1 1];
    A(i) = 0.5*sqrt(det([x;y;ons])^2 + det([y;z;ons])^2 + ...
        det([z;x;ons])^2);
end
area = A;

end
```

G. goodArea.m

```
function area = goodArea(triangulation,X,Y,Z,A,pointcloud,length,factor)
% goodArea determines the area of a meshed point cloud
% area = goodArea(triangulation,X,Y,Z,A,pointcloud,length,factor))
% - area is the area of the meshed point cloud
% - triangulation is the triangulated point cloud
% - X, Y, Z are the coordinates of the vertices of the triangles
% - A is the area of all triangles summed up
% - pointcloud is the pointcloud of which the triangulation is made
% - length is an integer; the maximum line length is Xlim/length
% - factor is an integer, indicating how much larger a triangle may be
% wrt the mean area


tri = triangulation;
sizetri = size(tri,1);
meanA = mean(A);
A_good = [];
W = []; % the Wth triangle is wrong
xlim = abs(pointcloud.XLimits(2)-pointcloud.XLimits(1))/length;

for i = 1: sizetri
    tri_num = i;   % triangle #
    % look at the length of the triangle edges
    % if one edge is longer than half of the lesion, it is a wrong triangle

    % tx is the x coordinates of the three points (3x1 matrix)
    tx = X(tri(tri_num,:));
    ty = Y(tri(tri_num,:));
    tz = Z(tri(tri_num,:));

    point1 = [tx(1,:),ty(1,:),tz(1,:)]; %xyz coordinates of point 1
    point2 = [tx(2,:),ty(2,:),tz(2,:)];
    point3 = [tx(3,:),ty(3,:),tz(3,:)];

    % calculate length of triangle edges
    % d = ((x2 - x1)2 + (y2 - y1)2 + (z2 - z1)2)^1/2
    length12 = (sqrt((point2(1)-point1(1))^2+(point2(2)-point1(2))...
        +(point2(3)-point1(3))^2));
    length23 = (sqrt((point3(1)-point2(1))^2+(point3(2)-point2(2))...
        +(point3(3)-point2(3))^2));
    length13 = (sqrt((point3(1)-point1(1))^2+(point3(2)-point1(2))...
        +(point3(3)-point1(3))^2));

    if A(i) > meanA*factor  % means the triangle does not belong
       if length12 > xlim || length23 > xlim || length12 >  xlim
            W(end+1) = i;
       end
    else
        A_good(end + 1) = A(i);
    end

end
area = A_good;

% plot wrong triangles
% get their coordinates
tx = X(tri(W,:));
ty = Y(tri(W,:));
tz = Z(tri(W,:));
% plot specific triangles
txc = [tx, tx(:,1)];       %close them for plotting
tyc = [ty, ty(:,1)];
tzc = [tz, tz(:,1)];
fill3(txc.', tyc.', tzc.', 'r');
hold off
end
```

H. erythemaScore.m

```
function intensity = erythemaScore(pointcloud,colour)
% erythemaScore determines the erythema intensity of the lesion point cloud
% intensity = erythemaScore(pointcloud,colour)
% - intensity is an integer between 1 and 4
% - pointcloud is the input point cloud of the lesion, in Y Cb or Cr
% - colour is one of the YCbCr components, same as the lesion point cloud

% look at whether the values of the point cloud colors are in one of the
% four intensities

figure;
for i = 1 : 4
    % read reference images
    EryRGB{i} = imread(sprintf('erythema_%s.PNG', num2str(i)));
    % transform them into ycbcr colour space
    YCBCR = rgb2ycbcr(EryRGB{i});
    COLOUR{i} = YCBCR(:,:,colour); % either y cb or cr
    % look at the maximum values
    MAX = max(max(COLOUR{i}));
    C = double(COLOUR{i} > MAX-60);
    CMAX{i} = uint8(C .* double(COLOUR{i}));
    subplot(2,2,i); imshow(CMAX{i});
    CM = CMAX{i};
    range{i} = [min(CM(CM > 0)),max(CM(CM > 0))];
end

erythema1 = [];
erythema2 = [];
erythema3 = [];
erythema4 = [];

range1 = range{1};
range2 = range{2};
range3 = range{3};
range4 = range{4};

for k = 1:size(pointcloud.Color,1)  % for every point in the pointcloud
    RED = pointcloud.Color(k,3);

    % if the colour of the point falls into the range
    if RED ≥ range1(1) && RED ≤ range1(2)
        erythema1(end+1) = 1;
    end
    if RED ≥ range2(1) && RED ≤ range2(2)
        erythema2(end+1) = 1;
    end
    if RED ≥ range3(1) && RED ≤ range3(2)
        erythema3(end+1) = 1;
    end
    if RED ≥ range4(1) && RED ≤ range4(2)
        erythema4(end+1) = 1;
    end

end
% determine which index has the highest value
summ = [sum(erythema1),sum(erythema2),sum(erythema3),sum(erythema4)];
[M,I] = max(summ);
intensity = I;
end
```

I. *intensity.m*

```
function IArea = intensity(F)
% intensity determines the intensity score of the area in percentages
% IArea = intensity(F)
% IArea = intensity score of the area (1-4)
% F = input percentage of lesion area to body area
if F < 90
    if F < 70
        if F < 50
            if F < 30
                if F < 10
                    if F < 1
                        IArea = 0;
                    else
                        IArea = 1;
                    end
                else
                    IArea = 2;
                end
            else
                IArea = 3;
            end
        else
            IArea = 4;
        end
    else
        IArea = 5;
    end
else
    IArea = 6;

end
end
```

J. pasi.m

```
function score = pasi(head,arms,trunk,legs)
% pasi calculates the PASI score from input intensity scores
% score = pasi(head,arms,trunk,legs)
% - score is the calculated PASI score
% - head is an array containing the intensity scores of the four criteria
% head = [area,erythema,induration,desquamation]
% - arms is the same as head, but then for the arms
% - trunk is the same as head, but then for the trunk
% - legs is the same as head, but then for the legs

% B1 = A1 * 0.1;
% B2 = A2 * 0.2;
% B3 = A3 * 0.3;
% B4 = A4 * 0.4;
%
% C1 = B1 * Area;
% C2 = B2 * Area;
% C3 = B3 * Area;
% C4 = B4 * Area;
%
% score = C1 + C2 + C3 + C4;

%head = [area,erythema,induration,desquamation]

A1 = sum(head(:,2:4));     % sum all intensities except area
A2 = sum(arms(:,2:4));
A3 = sum(trunk(:,2:4));
A4 = sum(legs(:,2:4));

B1 = A1 * 0.1;
B2 = A2 * 0.2;
B3 = A3 * 0.3;
B4 = A4 * 0.4;

C1 = B1 * head(1);
C2 = B2 * arms(1);
C3 = B3 * trunk(1);
C4 = B4 * legs(1);

score = C1 + C2 + C3 + C4;
end
```