Tuberculosis Detection by Purified Protein Injection Design Description Document

Sze Wah Lee (Project Coordinator)
Rebecca Silver (Customer Liaison)
Coco Yang (Document Handler)
Madilyn Beckman (Scribe)
Yi Wen Fan (Coding Expert)
Faculty Advisor: Prof. Andrew Berger

Document No. 001

Revisions Level	Date
D	February 26, 2018

This is a computer-generated document. The electronic master is the official revision. Paper copies are for reference only. Paper copies may be authenticated for specifically stated purposes in the authentication block.

Authentication Block

Contents

1 Revision History

Version	**Description**	**Date**	Authorization
A	Initial Draft DDD	23 Jan 2018	All
B	Revised Draft DDD	7 Feb 2018	All
C	Revised Draft DDD	21 Feb 2018	All
D	Midterm Review	26 Feb 2018	All
E	Revised Review	9 April 2018	All
F	Final Review	6 May 2018	All

2 Background

Tuberculosis is one of the top 10 causes of death worldwide. It is the leading cause of death in patients with HIV. This epidemic is especially prevalent in developing countries where access to testing and treatment is limited. Testing for tuberculosis is necessary because someone can be a carrier but not show signs, nor be contagious until they have immunosuppression. It is best to catch and treat tuberculosis before this point and this cheap testing is the best way to do that.

Latent TB is diagnosed by measuring the reaction of white cells that have responded to the presence of TB bacteria. One of the method is a skin test with PPD, Purified Protein Derivative. A volume of 0.1 ml PPD is placed in the skin of the patient.The reaction size of the skin area is then read after 48 to 72 hours, usually with a ruler to measure the width. This method dates back to 2000 years and often is dependent on the reader's judgement. The redness on the reaction spot also makes the measurement harder to see with the naked eye.

This senior design project goal is to improve the existing measuring method with optics and maintaining the cost low.

3 Vision Statement

The ideal system has a source and detector connected to a data processing device. The user would place the detector and source above the spot of skin with PPD injection. The device would be able to calculate the surface topography of the bleb after injection, and provide a volume measurement in milliliters. This includes, cross sectional area and height of the bleb. After the patient returns for a follow up meeting 48 hr to 72 hr later, the device would be able to calculate the cross sectional area, most importantly the width perpendicular to the arm, of the physical reaction on the dermis. We are exploring a profilometry method, an approach that meets the basic criteria requested by the customer. This approach explores the physical traits such as area and volume of the skin reaction to the PPD injection. This solution is similar to that of the current medical procedure, but will eliminate the subjectivity of measuring the spot by hand or simply judging size via the human eye.

4 Project Scope

We are responsible for creating a customized software to obtain data from our imaging device

1. A Matlab program that after analyzing images, provides width and volumetric data as well as a 3D model of the bleb or reaction spot for the technician.
2. A proof of concept setup: consisting mainly of a lens system imaging a line pattern and a camera to capture and record the image for processing
3. Explanation of next steps to take in order to achieve portability and easy to use requirements

4.1 Programming Theory

The equations below are from the paper [9.2.1.] and are the baseline for our program. These equations are very standard across many profilometry papers we read. A Matlab program was written to perform fourier transform, and phase unwrapping on the lab images. Numerical data was extracted from the shifts of the line pattern which is caused by the elevation of the bump. With this data, height and width of the reaction bump were calculated.

4.1.1 Forward Data Acquisition

$$g_o(x,y) = a(x,y) + b(x,y)Cos[2\pi f_o x + \phi_o(x)] \tag{1}$$

$$g_s(x,y) = a(x,y) + b(x,y)Cos[2\pi f_o x + \phi(x,y)] \tag{2}$$

$$G_s(u,v) = FFT(g_s); G_o^*(u,v) = FFT(g_o)^* \tag{3}$$

$$\Delta\phi(x,y) = \Im\{log(FFT^{-1}[G_s(u,v)G_o^*(u,v)])\} \tag{4}$$

$$h(x,y) = \frac{L_o}{\frac{2\pi L_o^2 d cos\theta}{P_o \Delta\phi (L_o + x cos\theta sin\theta)^2} - \frac{d cos\theta sin\theta}{L_o + x cos\theta sin\theta} + 1} \tag{5}$$

$$P_o = \frac{1}{f_o} = \frac{P}{cos\theta} \tag{6}$$

Figures 1 through 6 visualize the steps our algorithm follows. See the full code in the appendix.

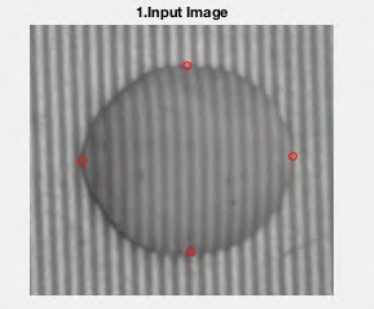


Figure 1: FInput image with boundary points (red) selected manually

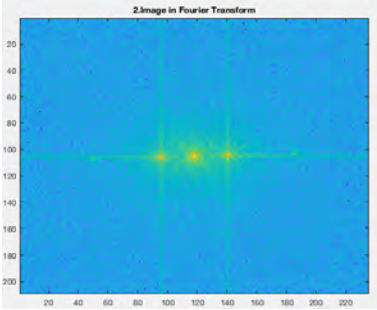


Figure 2: Image in Fourier Transform

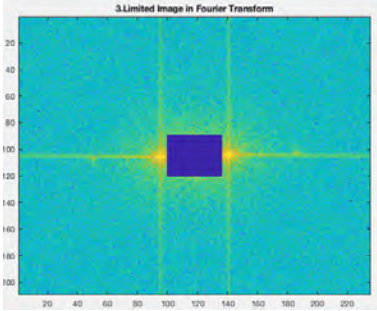


Figure 3: Limited image in Fourier Transform

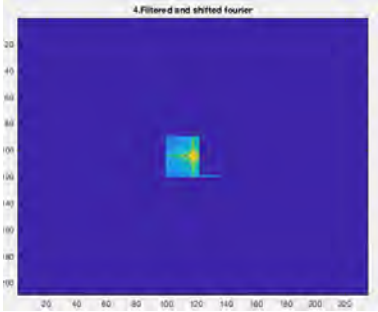


Figure 4: Filtered and shifted Fourier Transform

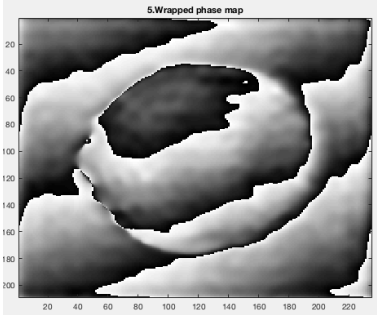


Figure 5: Wrapped phase map

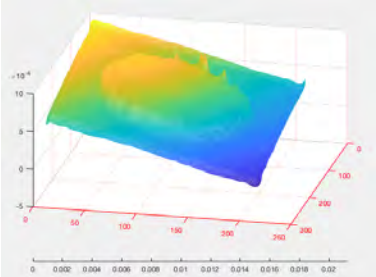


Figure 6: Result- A 3D model of the bump from figure 1

4.1.2 Width Measurement and Volume Calculation

Currently able to obtain the length, width and height of measured bump. This does require using human interaction with program to define boundaries. Next steps towards automation of boundary definition could involve training a neural network.

Though we are able to give fairly accurate measurements for the length, width and height of the reaction bump, we are not yet able to give a reliable value for the volume. This is due to the fact that our program cannot yet detect the boundary around the the bump. Since reaction bumps are not perfectly round or symmetric, we need information for the edge of the entire bump to deliver and accurate volume measurement. Currently, we are only able to manually select four points on the boundary of the bump and use the height at the center intersection, which is not an accurate representation of the volume of the bump. Once the edge detection algorithm is completed, calculating the an accurate volume will be possible.

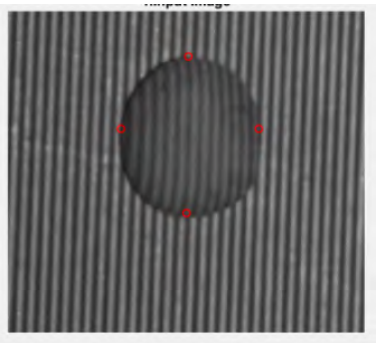


Figure 7: The technician needs to locate 4 points on the boundary of the bleb before the computer runs any calculations.

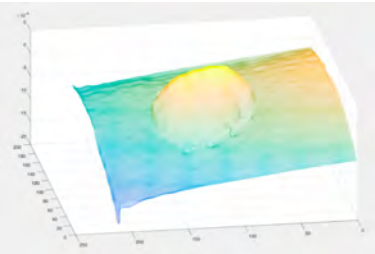


Figure 8: Best current results of bleb imaging.

Our current results for physical length measurements (completed with a ruler) versus using the program show promising results. The physical measurements should be measured with a more precise tool like calipers.

Physical Measurement	Computer Measurement	Percent Error
5.9 mm	5.4 mm	10.2%
6.3 mm	6.1 mm	3.2%
9.8 mm	9.7 mm	1.0%
9.6 mm	10.2 mm	6.3%

4.2 Experimental

4.2.1 Optical System Overview

The light source is collimated by the first lens , and evenly illuminates the grating that is placed in the path of light. Then the grating is imaged by a second lens at two focal lengths away, resulting in a magnification of 1. The image plane is placed perpendicular to the camera to prevent loss of information due to shadows. Two polarizers, placed after the grating and before the camera, improves images quality from light scattering off of the surface of the skin where the bleb would be measured.

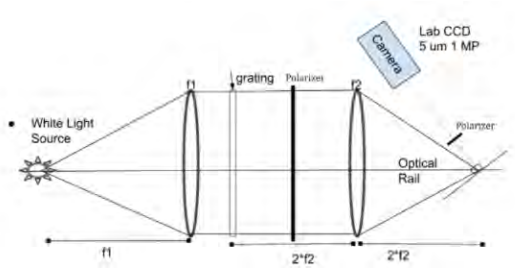


Figure 9: Schematics of the optical projection and detecting system. Grating is projected onto an image plane that is perpendicular to the camera.

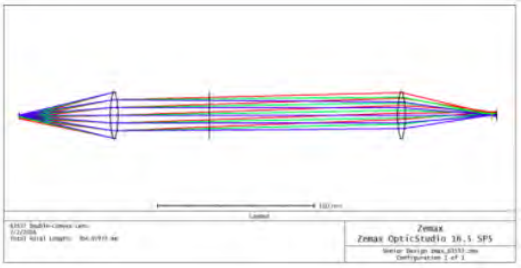


Figure 10: A Zemax stimulation of the design. The quality of the image of the grating was analyzed with CodeV. The MTF of the image is 0.5 at 10 lp/mm.

Illumination	System
Collimator Lens Diameter	30 mm
EFL	60 mm
F-Number	2
MTF	0.9 at 10 lp/mm
Magnification	1X
System Length	350 mm
Grating Spacing	2-4 lp/mm
Light Source	White LED, cell phone flashlight
CCD Resolution	1MP at 5 μm per pixel

4.2.2 Lab Setup

Our final presented system setup has two parallel polarizers [9.2.2] to reduce scatter; the image plane is perpendicular to the CCD instead of the projection system; the lens mounts were made smaller to reduce the angle between the projection and the CCD to 14.7°; and the system was rotated vertically to allow for patients to rest arm on flat surface under CCD.

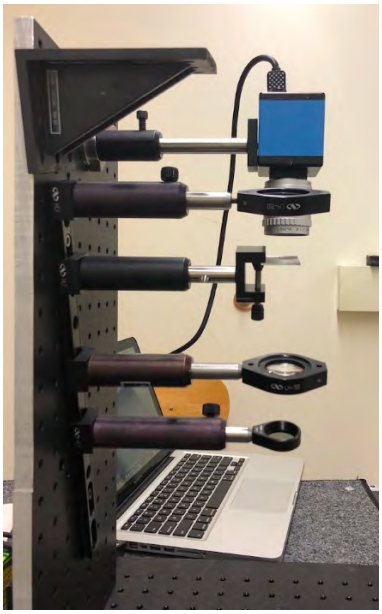


Figure 11: Final presented lab setup.

Figure 12: System from top view with CCD and projection arm.

4.2.3 Environment

As a device intended for clinics, it needs to operate in the following environment:

- Temperature: 59-77 °F – operation range
- Relative Humidity: 40% room temperature
- Outlet/battery power
- Ambient room light

5 In Vivo Testing

We were able to project the grating pattern onto skin and still maintaining fringe contrast despite scattering. A TB team member had a PPD injection from UHS, and attempted to measure the bleb with the designed system. However, due to an unexpected factor of sharing lab equipments, the system needed realignment. Therefore, the bleb dissipated before a measurement took place. Below are earlier images captured of a raised mole on the skin, this gives the same idea as what the injection would have been but only the same color.

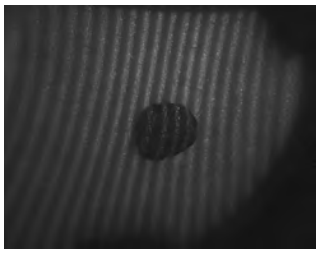


Figure 13: This is a mole on one of the teammates under green LED.

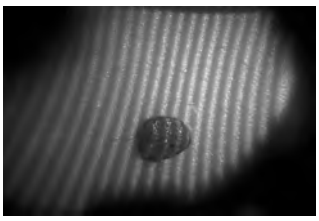


Figure 14: This is the same as Figure 13 but under red LED.

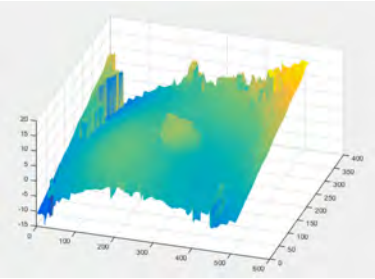


Figure 15: Results from the green light mole. This was early in our code and lab setup so it is not as high definition as we have been achieving. However it shows that skin testing does work.

6 Cost Analysis

This is the current cost analysis for what was presented on Design Day. The fourth colomn is the cost of the item had we not had the benefit of the teaching lab supplies. [9.2.3-4]

Items Needed	Item Found	Cost	
Visible Light Source	Cell Phone Flashlight	Free	-
Line Grating	Teaching Lab Equipment	Free	$150
Two Lenses	Edmund Optics (60mm FL, 30 mm Dia.)	$30 ea.	-
Camera	The Imaging Source, S/N:DMK31BF03	Free	TBD
Optical Mounts	Teaching Lab Equipment	Free	$30 ea.
Linear Polarizers	Teaching Lab Equipment	Free	$10-20 ea.

7 Design Day

Our plan was to present a system that when a created 3D object is placed in the system we will receive a computer 3d model and the volume, height and width measurements. Ideally we would have liked to use skin or chicken skin,but that proved to be difficult with the current equipment we have so we settled for a demo with our calibration bump.

At design day, we presented a vertically aligned prototype system to show proof of concept. People were able to put their arm in the system with a fake* (maybe a different word here) bump placed on their forearm. We captured that image and manually ran it through our program to produce a 3D model as well as the height, length and width of the bump.

We expanded our demo further by explaining our vision for the future of this prototype. We discussed how we used the cell phone light as the light source because our final vision is to have a small system that can be attach to and be used with a cell phone.

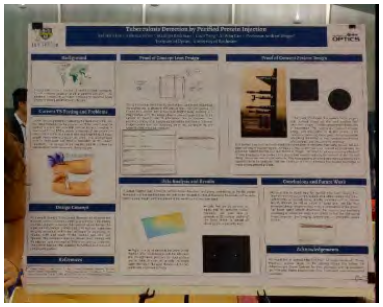


Figure 16: Our poster on Design Day May 4, 2018.

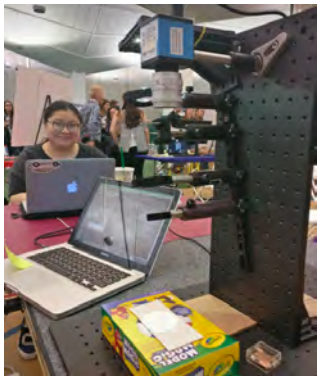


Figure 17: Our setup on Design Day May 4, 2018.

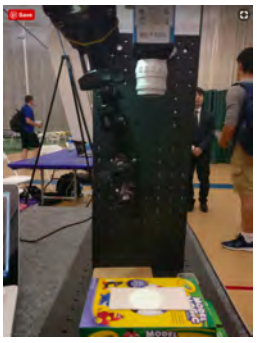


Figure 18: Our setup on Design Day May 4, 2018.

8 Future Design Suggestions

Truncate the System: One of the concerns of the customer was to have a design that is easily portable as well as small in size. We have worked towards achieving this goal with the reduction of angle between the projection and imaging arm, thus reducing the width of the system. The length of the design was limited by the focal length of the lens, 30 mm. As part of this project, we also created a truncated version of our design system with a goal of maintaining the field of view of an 25 mm object as well as an projected grating image of 1X magnification.

A ThorLab Achromatic Doublet (focal length of 25mm and a diameter of 12.7 mm) was used to attempt to replicate the performance of the imaging lens, lens between the grating and the image plane. The collimating lens was replaced by a Edmund Optics' double-convex lens with a focal length of 20 mm and a diameter of 15 mm. Although the new design's overall length was potentially truncated to 160.5 mm, almost half of the original design, the magnification was lowered to 0.29 to compensate for the 25 mm object field of view.

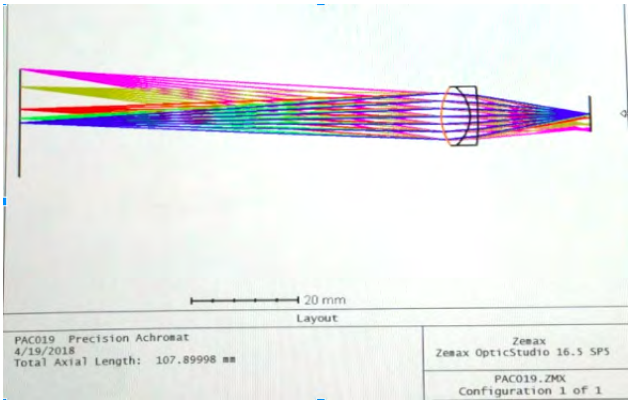


Figure 19: A Zemax simulation of a ThorLab Achromatic lens imaging an 25 mm diameter object, size of grating.[9.2.5]

Automation of Program: Ideally the program would be fully automated after an image is taken. In order to achieve this the program needs to be able to detect the boundaries of the bump without interaction. This can be achieved by training a neural network algorithm.

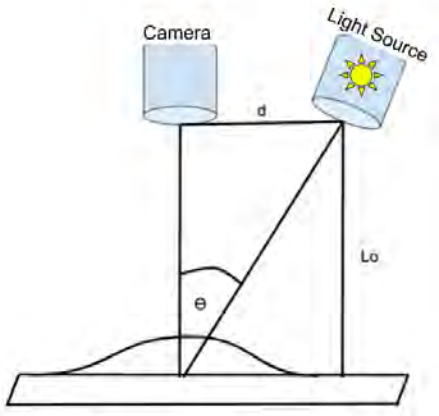
Cell Phone Attachment: One purpose of us using the cell phone light was to take the first step in proving this system could be isolated to a cell phone. Using a clip on projection system (to be designed) on the light source, the camera and an app. This would be the ultimate portability method. There is already a small angle between the light source and camera, however this will require some serious engineering and is possibly a project for another senior design team.

Reaction Spot Database: This technology would allow for a creation of a database that could give countless examples of reactions that were positive, negative, false-positive or false-negatives. This could lead to further understanding of the development of the disease.

9 Appendix

9.1 Variables

θ: angle between light source and camera
f_o: fundamental frequency of observed grating image
d: distance between light source and camera
P: Projected grating period
L_o: length between reference plane and image/object plane
G_o^*, G_s: Fourier transforms of original images



9.2 Citations

1. **Mathematic Theory**: Tavares, Paulo J., and Mario A. Vaz. "Linear Calibration Procedure for the Phase-to-Height Relationship in Phase Measurement Profilometry." *Optics Communications*, vol. 274, no. 2, 2007, pp. 307–314. 2007.02.038.
2. **Polarizer Theory**: Anderson, R. Rox. "Polarized Light Examination and Photography of the Skin." *JAMA Internal Medicine*, American Medical Association, 1 July 1991.
3. **Camera**: https://www.theimagingsource.com/products/industrial-cameras/firewire-400-monochrome/dmk31bf03/
4. **Collimating Lens**: https://www.edmundoptics.com/optics/optical-lenses/double-convex-dcx-spherical-singlet-lenses/15mm-dia.-x-20mm-fl-uncoated-double-convex-lens/
5. **Doublet** https://www.edmundoptics.com/optics/optical-lenses/double-convex-dcx-spherical-singlet-lenses/15mm-dia.-x-20mm-fl-uncoated-double-convex-lens/

9.3 Schedule Spring 2018

1. January
   - Create the Matlab Code
   - Collect lab equipments and make customized fringe patterns
2. February
   - Finish the Matlab Code
   - Design a 3D print Bump
   - Run Code in the lab with the experiment set up
3. March
   - Make Code revision
   - IA Presentation

4. April
    - Continue test trial
    - Expand test bump samples
    - Portability & Design Day
5. May
    - Final Document Submission

9.4 Progression of Image Results

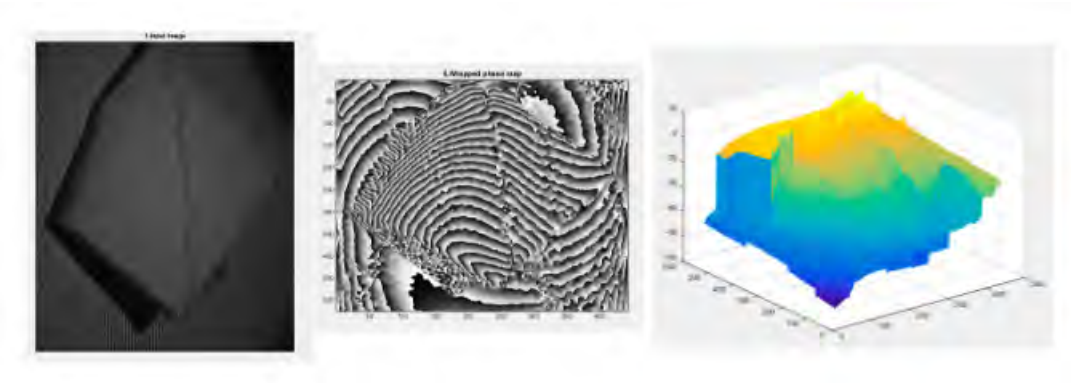


Figure 20: Feb 25. Paper. Grating 4 lp/mm. No Polarizers. Image perpendicular to source

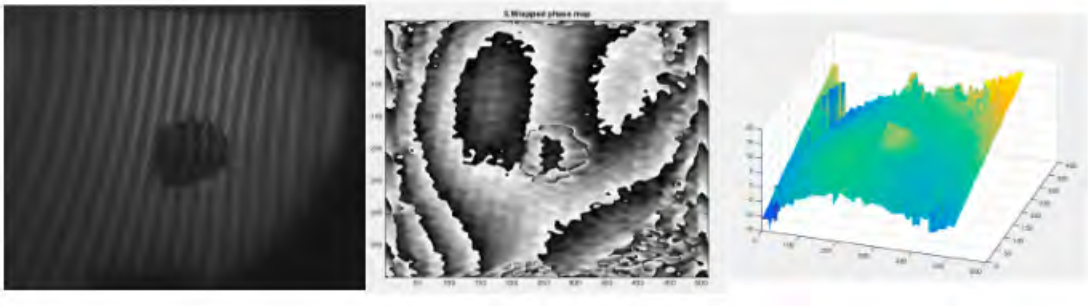


Figure 21: February 25. Mole. No polarizers. Grating: 1 lp/mm. Image Plane perpendicular to source.

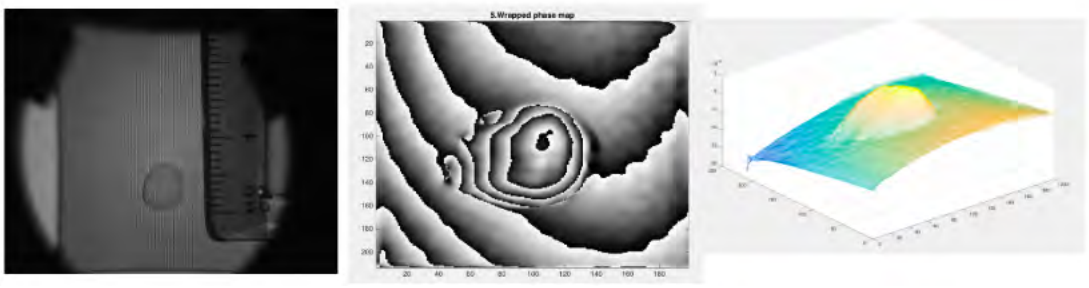


Figure 22: March 25,2018. Model Magic White Bump. 5 mm width on white background. Image Plane perpendicular to source. No polarizers

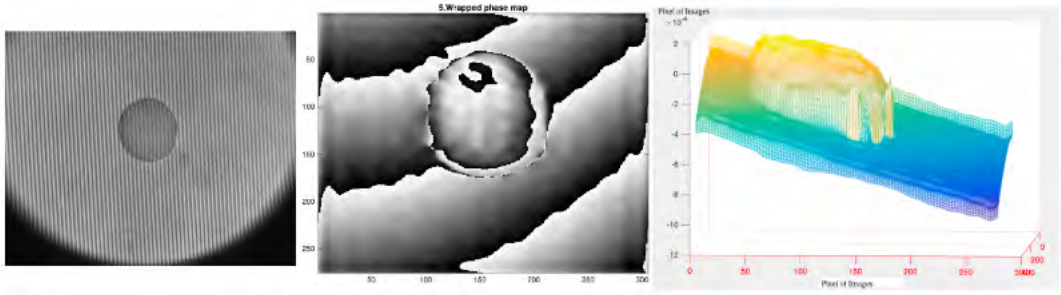


Figure 23: May 3. Bump 6 mm width on white background. With Polarizers and image plane perpendicular to camera.

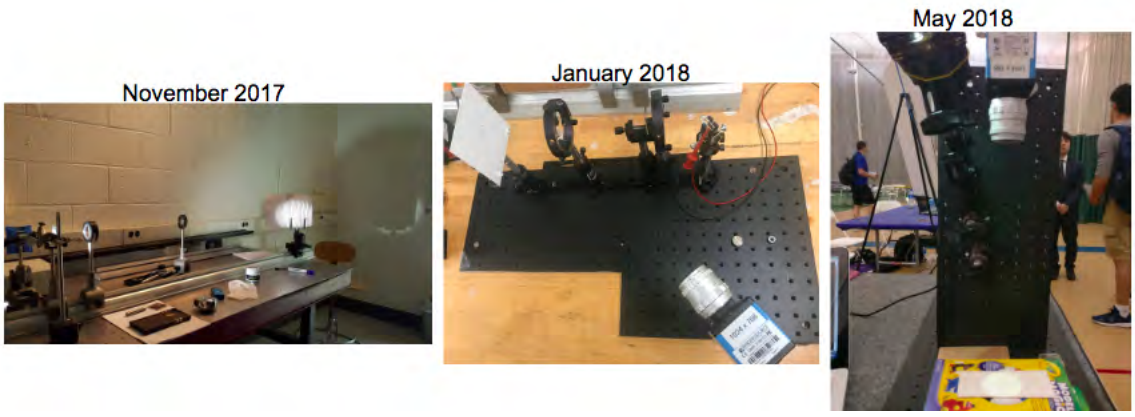


Figure 24: Development of lab setup.

9.5 Matlab Code

Contents


```
function p =unwrapping(g)
```

Initialization

```
[SX SY]=size(g);
MaxQueueSize=SX+SY;%maximum Queue Size
HalfMaxQueueSize=fix(MaxQueueSize/2);%half of MaxQueueSize
MinQualityThresh=0.01;%Minimum quality threshold
m=ones(SX,SY);

a=abs(g);%a=a.*mask;
p=angle(g);%phase to be unwrapped
L=bwlabel(m);
Lnumber=max(max(L));

Unwrapped=zeros(SX,SY);%to indicate whether a pixel has been unwrapped

Qa=zeros(MaxQueueSize,1);%amplitude (array for quened pixels,AQP)
Qx=zeros(MaxQueueSize,1);%x coordinate
Qy=zeros(MaxQueueSize,1);%y coordinate
Qn=0;%number of queued pixels

Pa=zeros(SX*SY,1);%amplitude (array for postponed pixels, APP)
Px=zeros(SX*SY,1);%x coordinate
Py=zeros(SX*SY,1);%y coordinate
Pn=0;%number of postponed pixels
Un=0; %number of pixels unwrapped
```

```
Not enough input arguments.

Error in unwrapping (line 4)
[SX SY]=size(g);
```

```
for i=1:Lnumber
    %find highest quality
    [start_x start_y]=find(a==max(max(a.*(L==i))),1,'first');
    %push into AQP
    [Qx Qy Qa Qn]=InsertQueue(Qx,Qy,Qa,Qn,start_x,start_y,a(start_x,start_y));
    Unwrapped(start_x,start_y)=1;%seed is taken as unwrapped
    Un=Un+1;%update Un

end
```

for each quened pixel, its four neighbors are unwrapped.

```
while Qn>0
    %step 3.1:
    cx=Qx(1);cy=Qy(1);

    Qx(1:Qn-1)=Qx(2:Qn);%delete
    Qy(1:Qn-1)=Qy(2:Qn);%delete
    Qa(1:Qn-1)=Qa(2:Qn);%delete
    Qn=Qn-1;

    %push the left neighbor into the AQP or APP
    if cx-1>0 && Unwrapped(cx-1,cy)==0 && m(cx-1,cy)==1
        %unwrapping the left neighbor
        p(cx-1,cy)=p(cx-1,cy)-round((p(cx-1,cy)-p(cx,cy))/2/pi)*2*pi;
        if a(cx-1,cy)>MinQualityThresh %push into AQP if quality is high
            [Qx Qy Qa Qn]=InsertQueue(Qx,Qy,Qa,Qn,cx-1,cy,a(cx-1,cy));
            if Qn==MaxQueueSize %if AQP reaches preset size, split it
                [Qn,Px,Py,Pa,Pn,MinQualityThresh]= SplitQueue(Qx,Qy,Qa,Qn,Px,Py,Pa,Pn,HalfMaxQueueSize);
            end
        else %push into APP if qualtiy is low
            Pn=Pn+1;Px(Pn)=cx-1;Py(Pn)=cy;Pa(Pn)=a(cx-1,cy);
        end
        Unwrapped(cx-1,cy)=1;%mark this pixel as unwrapped.
        Un=Un+1; %update Un

    end
    %push the right neighbor into the AQP or APP
    if cx+1<SX+1 && Unwrapped(cx+1,cy)==0 && m(cx+1,cy)==1
        p(cx+1,cy)=p(cx+1,cy)-round((p(cx+1,cy)-p(cx,cy))/2/pi)*2*pi;
        if a(cx+1,cy)>MinQualityThresh
            [Qx Qy Qa Qn]=InsertQueue(Qx,Qy,Qa,Qn,cx+1,cy,a(cx+1,cy));
            if Qn==MaxQueueSize
                [Qn,Px,Py,Pa,Pn,MinQualityThresh]= ...
                    SplitQueue(Qx,Qy,Qa,Qn,Px,Py,Pa,Pn,HalfMaxQueueSize);
            end
        else
            Pn=Pn+1;Px(Pn)=cx+1;Py(Pn)=cy;Pa(Pn)=a(cx+1,cy);
        end
        Unwrapped(cx+1,cy)=1;
        Un=Un+1;

    end
    %push the upper neighbor into the AQP or APP
    if cy-1>0 && Unwrapped(cx,cy-1)==0 && m(cx,cy-1)==1
        p(cx,cy-1)=p(cx,cy-1)-round((p(cx,cy-1)-p(cx,cy))/2/pi)*2*pi;
        if a(cx,cy-1)>MinQualityThresh
            [Qx Qy Qa Qn]=InsertQueue(Qx,Qy,Qa,Qn,cx,cy-1,a(cx,cy-1));
            if Qn==MaxQueueSize
                [Qn,Px,Py,Pa,Pn,MinQualityThresh]= ...
                    SplitQueue(Qx,Qy,Qa,Qn,Px,Py,Pa,Pn,HalfMaxQueueSize);
            end
        else
            Pn=Pn+1;Px(Pn)=cx;Py(Pn)=cy-1;Pa(Pn)=a(cx,cy-1);
        end
        Unwrapped(cx,cy-1)=1;
        Un=Un+1;

    end
    %push the lower neighbor into the AQP or APP
```

```
    if cy+1<SY+1 && Unwrapped(cx,cy+1)==0 && m(cx,cy+1)==1
        p(cx,cy+1)=p(cx,cy+1)-round((p(cx,cy+1)-p(cx,cy))/2/pi)*2*pi;
        if a(cx,cy+1)>MinQualityThresh
            [Qx Qy Qa Qn]=InsertQueue(Qx,Qy,Qa,Qn,cx,cy+1,a(cx,cy+1));
            if Qn==MaxQueueSize
                [Qn,Px,Py,Pa,Pn,MinQualityThresh]= ...
                    SplitQueue(Qx,Qy,Qa,Qn,Px,Py,Pa,Pn,HalfMaxQueueSize);
            end
        else
            Pn=Pn+1;Px(Pn)=cx;Py(Pn)=cy+1;Pa(Pn)=a(cx,cy+1);
        end
        Unwrapped(cx,cy+1)=1;
        Un=Un+1;

    end

    %if AQP is empty, copy data from APP
    if Qn==0 && Pn>0
        [Qx,Qy,Qa,Qn,Px,Py,Pa,Pn,MinQualityThresh]= ...
            Copy2Queue(Qx,Qy,Qa,Qn,Px,Py,Pa,Pn,HalfMaxQueueSize);
    end
end
p=p.*m+(min(min(p.*m))-2*pi).*(1-m);
```

Insert a pixel into AQP

```
function [Qx, Qy, Qa, Qn]=InsertQueue(Qx,Qy,Qa,Qn,x,y,a)
```

```
I=find(Qa(1:Qn)<a,1,'first'); %find its proper inserting point
if isempty(I) %put in the end of AQP
    Qx(Qn+1)=x;
    Qy(Qn+1)=y;
    Qa(Qn+1)=a;
else %inset into AQP
    Qx(I+1:Qn+1)=Qx(I:Qn);
    Qx(I)=x;
    Qy(I+1:Qn+1)=Qy(I:Qn);
    Qy(I)=y;
    Qa(I+1:Qn+1)=Qa(I:Qn);
    Qa(I)=a;
end
Qn=Qn+1;%update Qn
```

Split the Queue if it is too long

```
function [Qn,Px,Py,Pa,Pn,MinQualityThresh]= ...
    SplitQueue(Qx,Qy,Qa,Qn,Px,Py,Pa,Pn,HalfMaxQueueSize)
```

```
%put second half of AQP into APP
Pa(Pn+1:Pn+Qn-HalfMaxQueueSize)=Qa(HalfMaxQueueSize+1:Qn);
Px(Pn+1:Pn+Qn-HalfMaxQueueSize)=Qx(HalfMaxQueueSize+1:Qn);
Py(Pn+1:Pn+Qn-HalfMaxQueueSize)=Qy(HalfMaxQueueSize+1:Qn);
Pn=Pn+Qn-HalfMaxQueueSize; %update Pn
Qn=HalfMaxQueueSize; %update Qn
MinQualityThresh=Qa(Qn);%Update MinQualityThresh
```

Copy pixels from APP to AQP if AQP is empty

```
function [Qx,Qy,Qa,Qn,Px,Py,Pa,Pn,MinQualityThresh]= ...
    Copy2Queue(Qx,Qy,Qa,Qn,Px,Py,Pa,Pn,HalfMaxQueueSize)
Cn=min(Pn,HalfMaxQueueSize);%number of pixel to be copied
[temp I]=sort(Pa(1:Pn),'descend'); %sort APP and store in 'temp'
Qa(1:Cn)=temp(1:Cn); %copy to AQP
Qx(1:Cn)=Px(I(1:Cn));
Qy(1:Cn)=Py(I(1:Cn));
Qn=Cn; % update Qn
MinQualityThresh=Qa(Qn); %update MInQualityThresh
Pa(1:Pn-Cn)=temp(Cn+1:Pn); %arrange APP
Px(1:Pn-Cn)=Px(I(Cn+1:Pn));
Py(1:Pn-Cn)=Py(I(Cn+1:Pn));
Pn=Pn-Cn;%update Pn
```

Published with MATLAB® R2017a

Contents

Clean all the setting

```
clear; clc;close all;
```

1. Load the image

```
%image = imread('coin111.tif'); % read an image
%image = imread('sample1.jpeg'); % read an image
%image = imread('gl_2.png');  % read an image
%image = imread('rl3.png');  % read an image
%image = imread('b&w.png');  % read an image
%image = imread('MM1.bmp');  % read an image
image = imread('biggerbumponwhite.bmp');  % read an image


%image = imread('1.png'); % read an image

if(size(image,3)>1)             % Color to Black&white
    image=rgb2gray(image);
end
image=im2double(image);         % Convert Image to Double Precision
imshow(image);title('1.Input Image')% display the original image
```

Distance

```
hold on;
xy = [];
n = 0;
distance = [];
while n < 4
    [xi,yi] = ginput(1);
    plot(xi,yi,'ro')
    n = n+1;
    xy(:,n) = [xi;yi];
end

%disp(xy(1,1));
%disp(xy(1,2));
%disp(xy(2,1));
%disp(xy(2,2));
disp('Pick 2 points for width and then 2 for length');
C_d = 16410;
d1= sqrt((xy(2,2)-xy(1,1))^2+(xy(1,2)-xy(2,1))^2);
d1= d1 / C_d;
d2= sqrt((xy(2,4)-xy(1,3))^2+(xy(2,3)-xy(1,4))^2);
```

```
d2= d2 / C_d;

disp('The width is: (m)');
disp(d1);
disp('The length is: (m)');
disp(d2);
```

```
Pick 2 points for width and then 2 for length
The width is: (m)
    0.0066

The length is: (m)
    0.0069
```

2. Fourier transform // https://www.mathworks.com/help/matlab/ref/fftshift.html

```
image_fft = fftshift(fft2(image));  %%// https://www.mathworks.com/help/matlab/ref/fft2.html
figure; imagesc(log(abs(image_fft))); %%// https://www.mathworks.com/help/matlab/ref/imagesc.html
title('2.Image in Fourier Transform');
index = zeros(size(image_fft));

[x, y] = find(abs(image_fft)==max(max(abs(image_fft))));
image_fft(x-15:x+15,y-18:y+18) = 0; %% select area with most information and least noise
figure; imagesc(log(abs(image_fft)));
title('3.Limited Image in Fourier Transform');
```

3. Filtering the spectrum centered around fo and translating to the origin results in

```
[side_max_x,side_max_y] = find(abs(image_fft)==max(max(abs(image_fft))),1,'first');
index(side_max_x-15:side_max_x+15,side_max_y-18:side_max_y+18) = 1;
image_fft_bpf=image_fft.*index;
shift_x = x - side_max_x;
shift_y = y - side_max_y;
image_fft_bpf=circshift(image_fft_bpf,[shift_x shift_y]);%% https://www.mathworks.com/help/matlab/ref/circshift.html
figure; imagesc(log(abs(image_fft_bpf))); title('4.Filtered and shifted fourier');

image_rec = ifft2(fftshift(image_fft_bpf));  %% Inverse transformation to the image
phi = atan(imag(image_rec)./real(image_rec));
figure; imagesc(phi);title('5.Wrapped phase map');colormap gray;
```

4. Phase Unwrapping using quality guided Flood Filling algorithm

```
image_unwrp = unwrapping(image_rec);
C = 8.5*10e-6; %% pixel to meter constant
image_unwrp = image_unwrp * 8.5*10e-6;
```

5. 3D calibration

```
figure;imagesc(image_unwrp);
figure;mesh(image_unwrp);   %% https://www.mathworks.com/help/matlab/ref/mesh.html
ax1 = gca;
ax1.XColor = 'r';
ax1.YColor = 'r';
```

```
ax1_pos = ax1.Position; % position of first axes
%ax2 = axes('Position',ax1_pos,'XAxisLocation','bot','YAxisLocation','left','Color','none');
%ax2 = axes('Position',[0.1 0.1 0.8 0.001],'Color','none')
first_axis = gca;
sqz = 0.12; %// distance to squeeze the first plot
set(first_axis, 'Position', get(first_axis, 'Position') + [0 sqz 0 -sqz ]);
ax2 = axes('Position', get(first_axis, 'Position') .* [1 1 1 0.001] - [0 sqz 0 0],'Color','none');
scale_factor = 8.5*10e-6; %// change this to your satisfaction
xlim(get(first_axis, 'XLim') * scale_factor);
set(ax2, 'XScale', get(first_axis, 'XScale'));
```

Show the highest and lowest point in this 3D model

```
disp(max(max(image_unwrp)));
disp(min(min(image_unwrp)));
```

```
    9.1143e-04

   -2.7599e-04
```

Published with MATLAB® R2017a