Paweł Wągrodzki
Łukasz Sarnacki

0. About project

Input parameters:

Pixel size of CCD: dx = 3.45
Transverse magnification: M = -10
Wavelength of light: lambda = 0.6328 um
RI of medium: n0 = 1
RI of glass: 1.45701

Program is fully automated and full code is given in 7. Name of file: **PROJECT_FINAL.m**

1. Extract the field using TPS formulas, bring the field to the focus

Firstly we applied TPS formulas for phase extraction. After that we applied autofocusing..

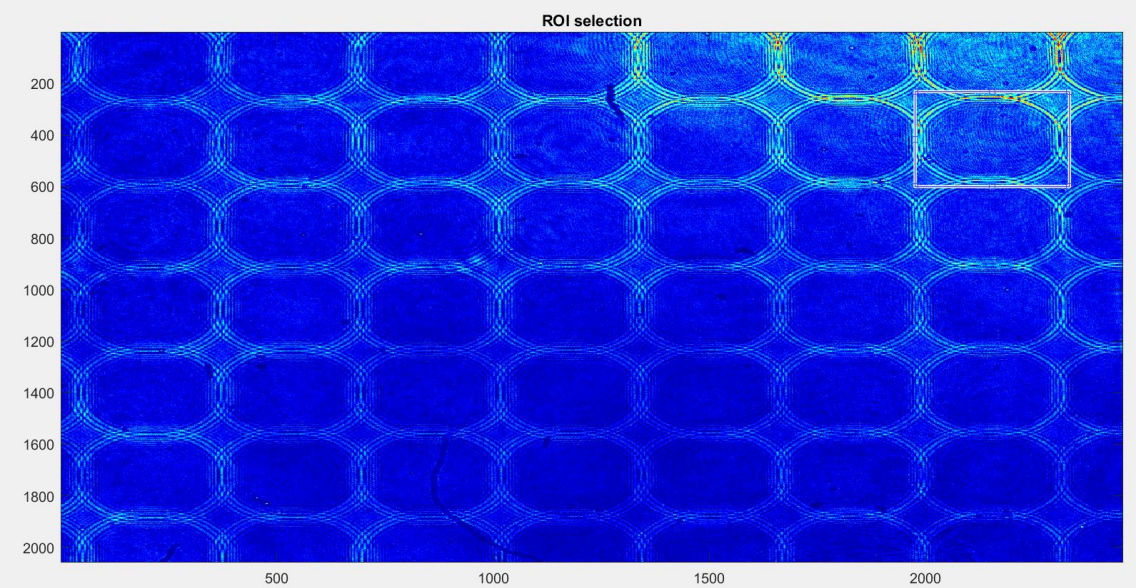


Fig. 1 Area with zero padding

First to see the shape of focus curve and where the focus field is the loop was with from -50 um to 150 um with 1 step . Then we change to smaller step near the minimum value (113 um ,so it was 100-130 with 0.1 step). The final value was 113.7 um.

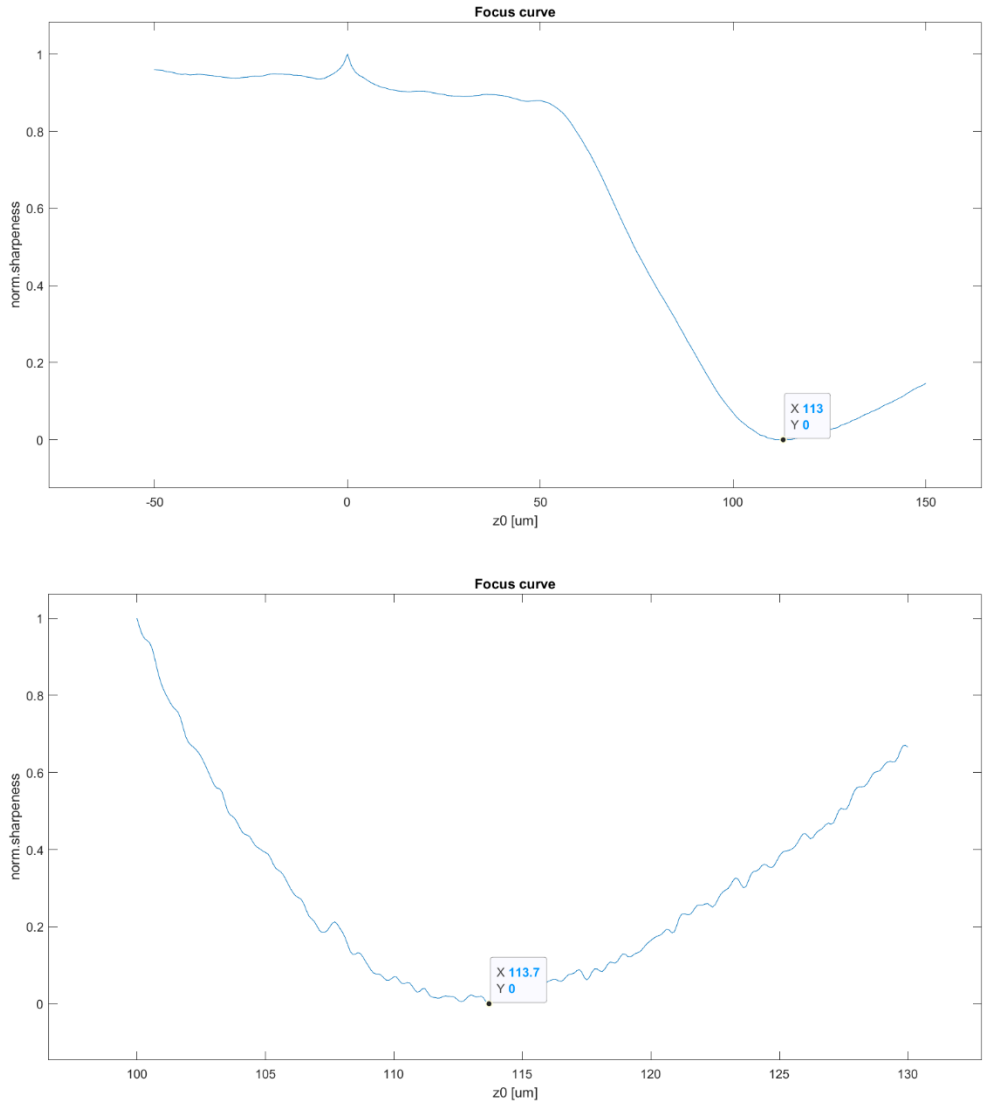


Fig. 2 Finding proper distance

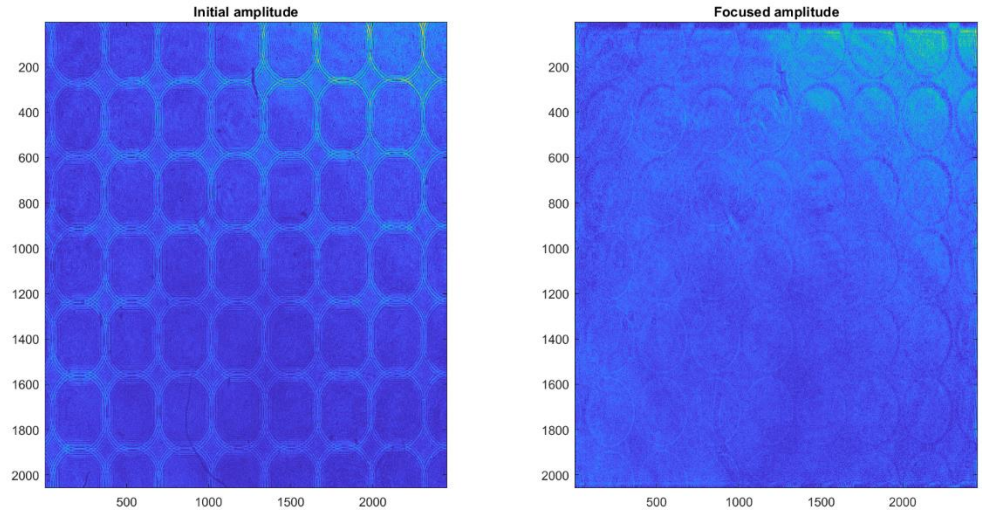


Fig. 3 Comparison of amplitudes

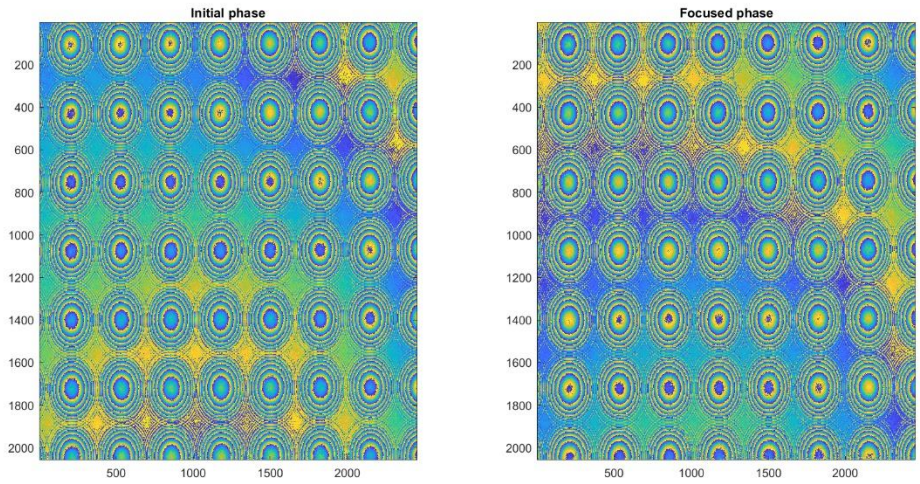


Fig. 4 Comparison of phases

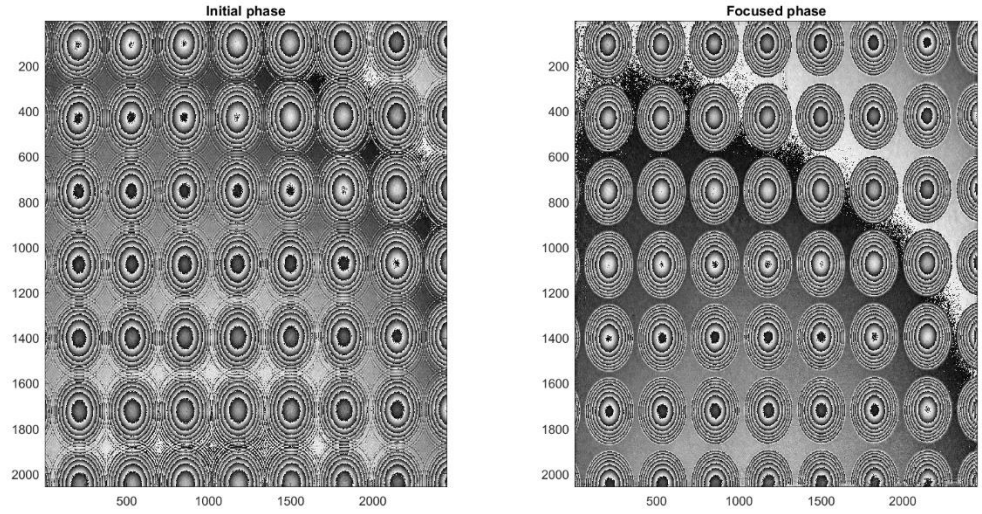


Fig. 5 Comparison of phases in grey scale

2. Unwrapping

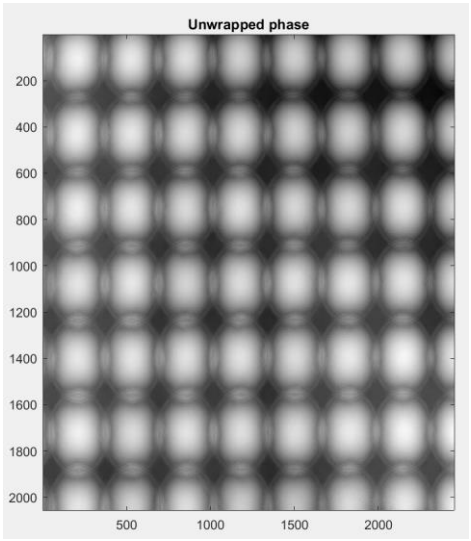


Fig. 6 Unwrapped phase from given function

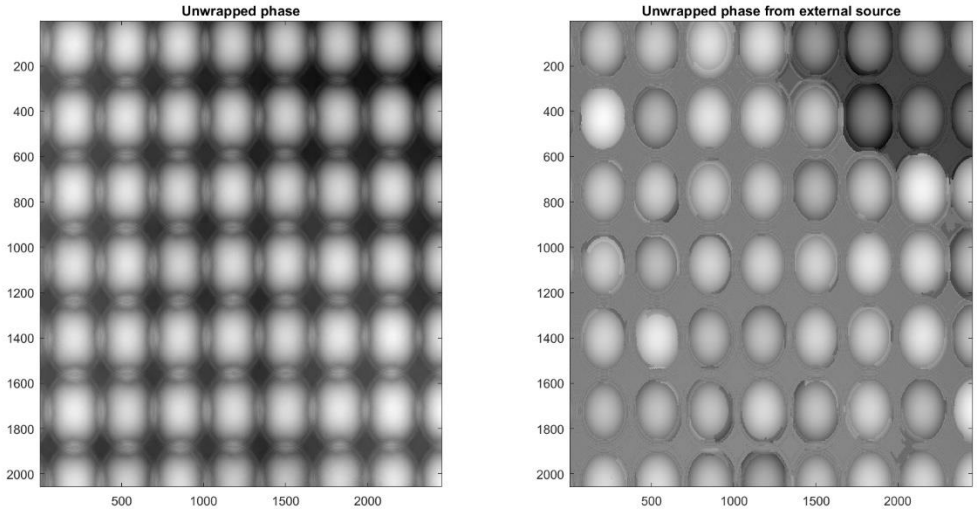


Fig. 7 Comparison of results from 2 different methods

Last form of unwrapping was using the unwrap function from MATLAB.

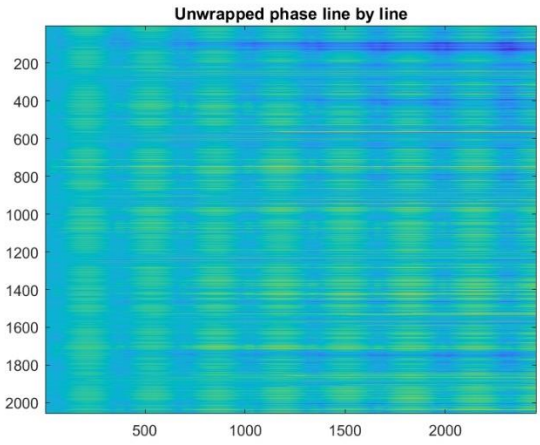


Fig. 8 Unwrapping line by line

To compare the results we check the RMS of each unwrapping. RMS was checking for each X-axis row. We also created matrix 2058 x 1 with values from 1 to 2058 to make a plot with RMS for every case. Code for that is given below:

```
%% RMS checking
rmse = rms(ph_before-unwrapped_1,2);
rmse_2 = rms(ph_before-unwrapped_2,2);
rmse_3 = rms(ph_before-unwrapped_3,2);
for y=1:1:length(rmse)
    pixels(y,1)=y;
end
pixels = linspace(1,length(rmse),length(rmse));
% Plots
figure
subplot(1,1,1)
title('RMS for 2 types of unwrapping') % changed from 3
plot(pixels,rmse,pixels,rmse_2,pixels,rmse_3)
xlabel('Y row')
ylabel('RMS')
legend({'given function','github','MATLAB unwrap'},'Location','northwest');

plot(pixels,rmse,pixels,rmse_2)
xlabel('Y row')
ylabel('RMS')
legend({'given function','github'},'Location','northwest');
```

Fig. 9 RMS calculating

```
%% unwrapping phase
% function TIE_unwrap2DCdct2

ph_1= angle(uout);
figure
unwrapped_1 = TIE_unwrap2DCdct2(ph_1,dx,lambda);
subplot 121
colormap gray
imagesc(unwrapped_1)
title('Unwrapped phase')

f = @() (TIE_unwrap2DCdct2(ph_1,dx,lambda));
times(1,1)=timeit(f);

% function unwrap from github TODO -> Source etc
%  Apply for loop

unwrapped_2=unwrap_phase(ph_1);
f = @()(unwrap_phase(ph_1));
times(1,2)= timeit(f);
subplot 122
colormap gray
imagesc(unwrapped_2)
title('Unwrapped phase from external source')
```

Fig.10 Part of code responsible for unwrapping

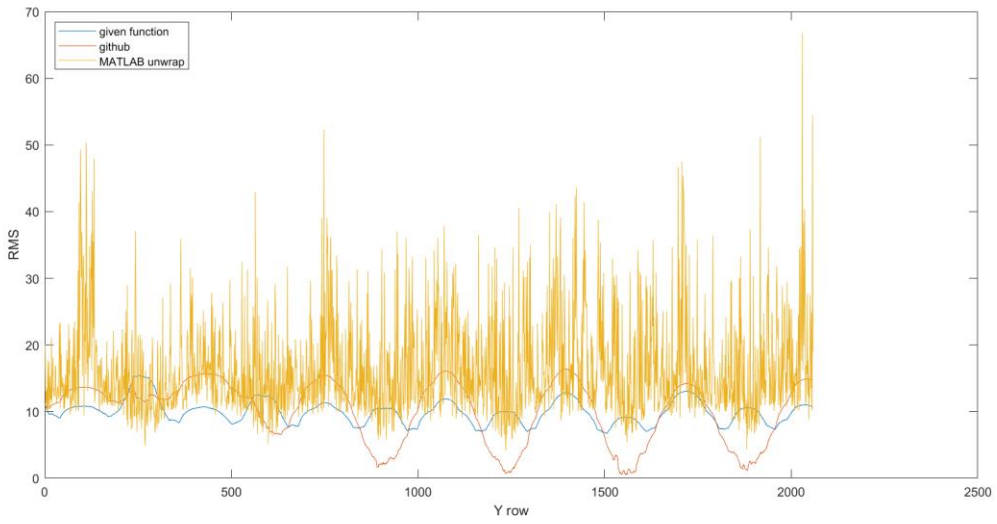


Fig. 11 Choosing the best unwrapping from RMS analysing

We can see that unwrap function give the biggest RMS. To compare 2 others we made another plot.

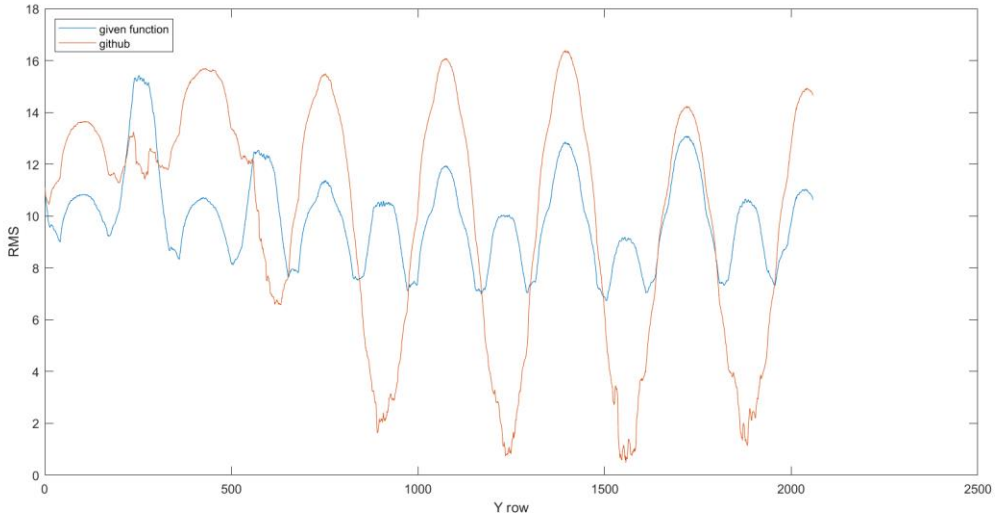


Fig. 12 Comparing RMS for 2 cases

We can see that RMS value depends on which row we are analysing. However in more cases RMS is bigger for GitHub function, and also in case of using given function we obtain to have almost unitary background. That is why we chose given function *TIE_unwrap2DCdct2* for unwrapping.

We also analyse the time of computation of functions

```
f = @()(unwrap(ph_1,[],2));
times(1,3) = timeit(f);
```

Figure 11 Part of code responsible for time measurement

Given function [s]	Github function[s]	Unwrap function[s]
2.8151979424	14.4283279424	0.3439364424

In case of time consuming unwrap function is the fastest however it gave wrong results – biggest RMS, not visible image. Given function *TIE_unwrap2DCdct2* is faster than function *unwrap_phase* from github,so that is another argument for choosing it .

For that function we applied binarization to obtain inhomogenous background.

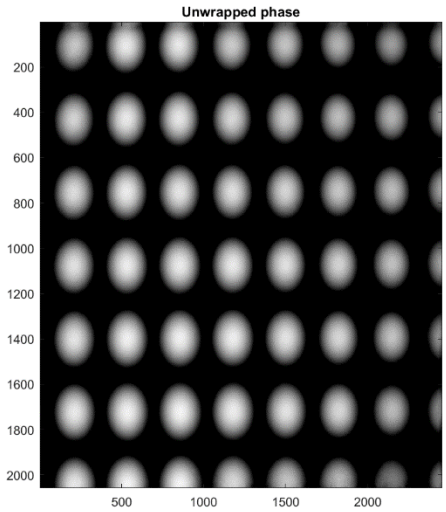


Fig. 12 Unwrapped phase after binarization

3. Circle detection

For circle detection we applied *imfindcircles* function from MATLAB. To We searched for radius for range (38,300).

```
% Circle detection
figure
subplot 111
colormap gray
unwrapped_1(1:end,2390:end)=0;
imagesc(unwrapped_1)
[centers, radii, metric] = imfindcircles(unwrapped_1,[38 300]);

viscircles(centers, radii,'LineStyle','--');
```

Fig. 13 Code responsible for circle detection

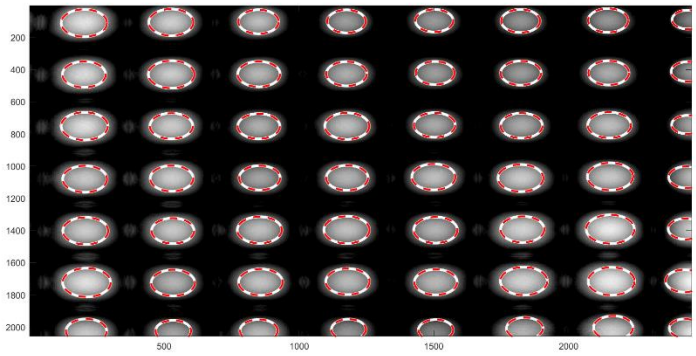


Fig. 14 Circle detection

4. Lens analysing

From that image we took a one lens to check its height from analysing the phase. Its coordinates was (363,262),(363,605),(707,262) and (707,605). We need to analyse phase of that lens so that is why we took the part of ***unwrapped image***.

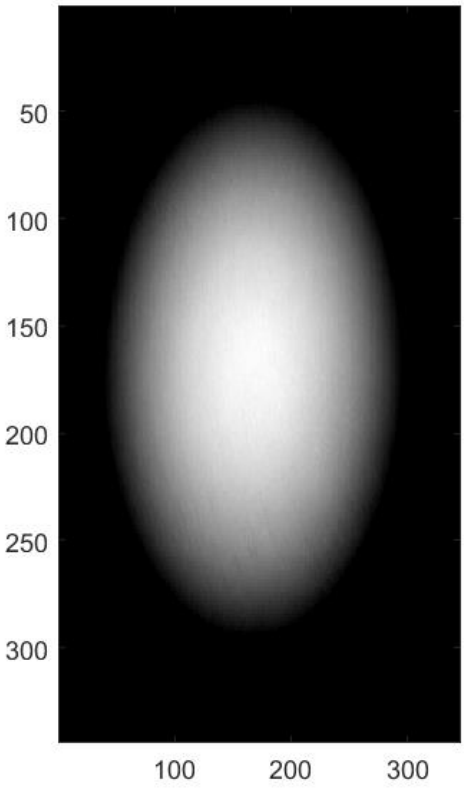


Fig. 15 Lens for calculating heigh

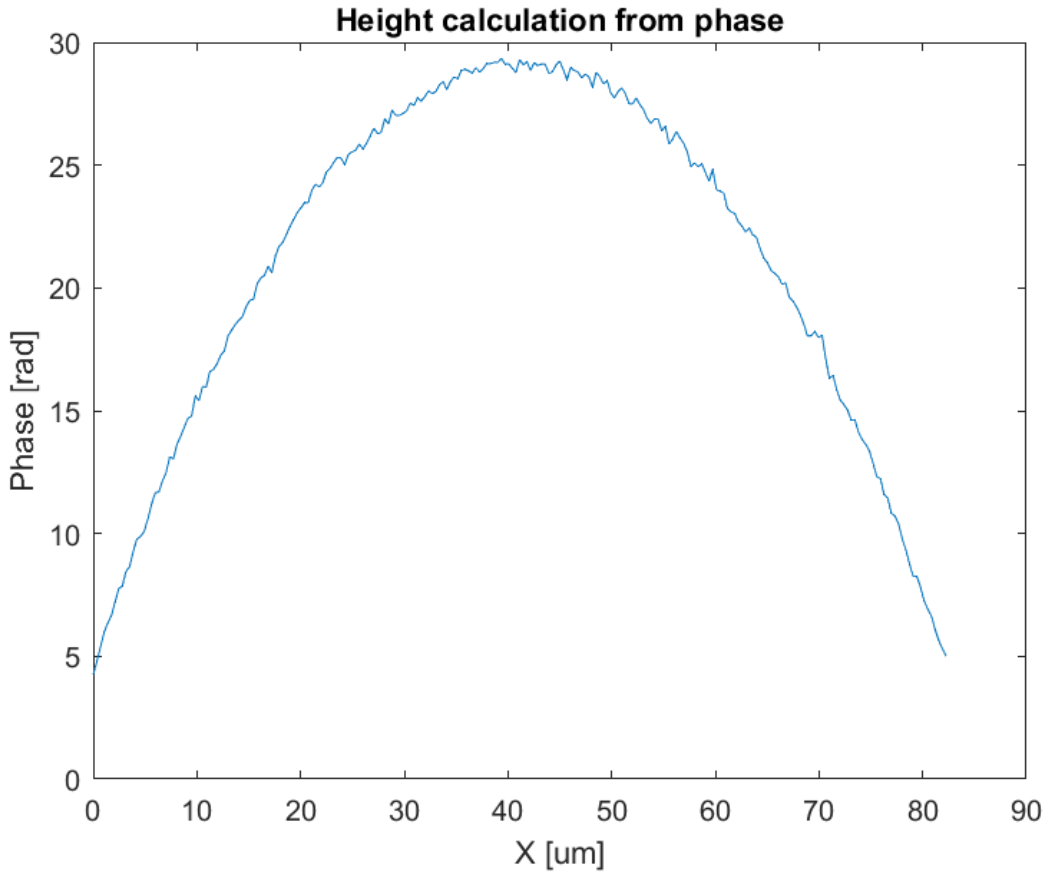


Fig. 16 Height calculations from phase

Maximum value was 29.46 and it was value of our chosen lens. From that value we calculated the heigh of lens from formula:

$$h = \frac{\varphi\lambda}{2\pi(n-1)}$$

where
λ = 0.6328 um
φ =20.26
n = 1.45701

So height:

From profile of phase, width of lens was determined which was about $2a$ = 88 um.
To be more precise we analysed the whole raw of lenses and take the biggest value

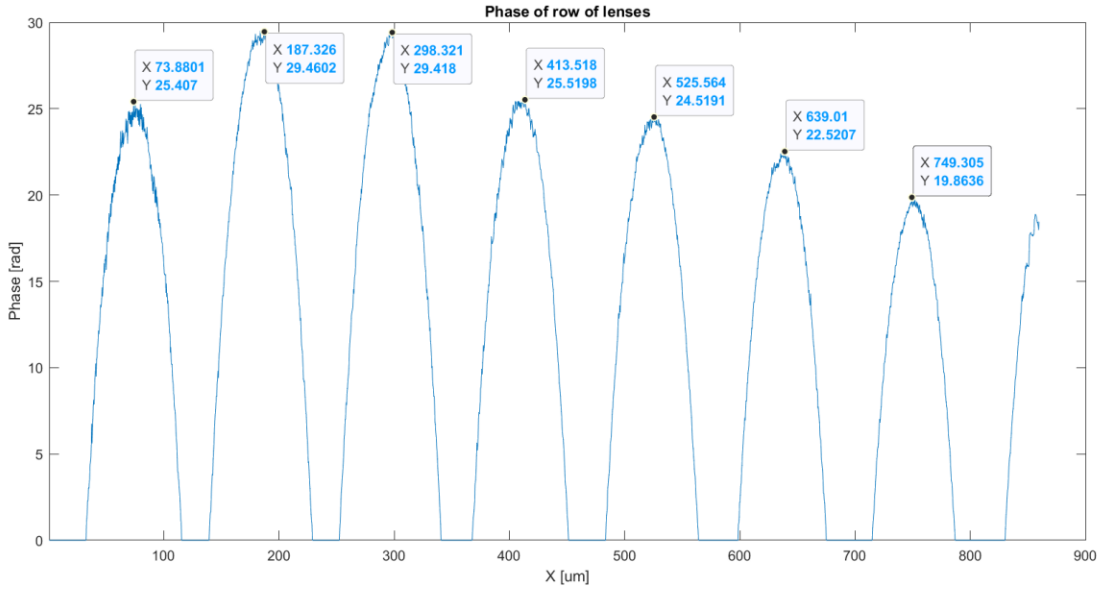


Fig. 17 Phase of row of lenses

To be sure that is proper value we searched from the max value from unwrapped phase. The max value was 31.79.

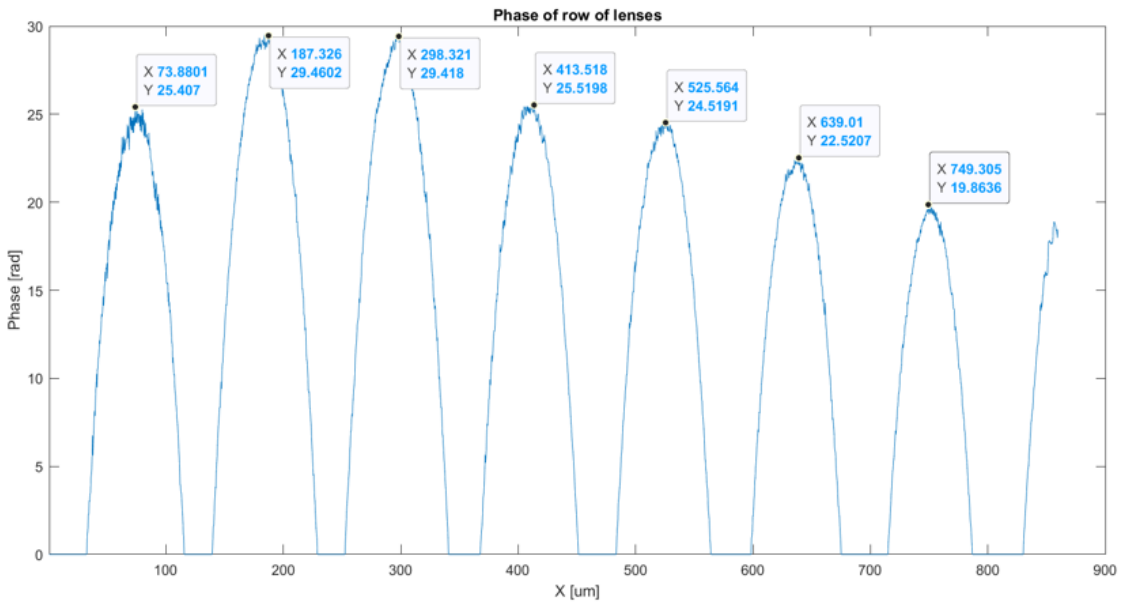
5. Calculated parameters

From given formulas we calculated parameters:

h [um]	d [um]	a [um]	NA	n	f [um]	ROC
7,010117	88	44	0,159321	1,45701	309,8208	141,5912

Order N°	Type	Pitch	ROC	Wave-length	Numerical Aperture	Working Distance	Array-Size (mm)	Price per Unit Euro
39 9901	FC-Q-100	100 µm	120 µm	UV – IR	0.19	240 µm	5 x 5	75,00
39 9905	FC-Q-100	100 µm	140 µm	UV – IR	0.17	280 µm	10 x 10	300,00
12-1640	FC-Q-250	250 µm	535 µm	UV – IR	0.11	1170 µm	10 x 10	300,00
09-1605	FC-Q-250	250 µm	580 µm	UV – IR	0.1	1270 µm	12 x 12	432,00

We can see that our parameters are correct. The biggest difference is in working distance, that error is about **(309,8208-280):280 ≈ 10%.** NA is similar (0,16 to 0,17). ROC is similar to that catalogue note. What is hard to decide was the d parameter, because the length of one lens is not equal to another, we decide the value of 90 um because it was a value of diameter of lens with the biggest phase value. Difference in values of diameter (90 to 100) can happen because of possible little missaligment of photos or having the value of zero in case of background (a small part of lens could had been counted as backround because of not enough value).



6. Summary

We managed to correctly achieve the main assumptions of the project. First, we extract the field using TPS formulas. Later in the unwrapped phase. The values depended on the row we analysed. Then we wrote a wheel detection program, it worked without a hitch. Phase analysis for one lens showed that the maximum value varied was 19,78,but to be more precise we applied averaging of the phase based on one row of lenses.
We had a slight difficulty in determining d, it is related to the arbitrariness in determining the beginning and end of the jumper. For this reason, depending on the designation, we will have a slightly different result.

In our results you can clearly see the Focus phase and also that there is focus.

The curvature of the background is due to optical aberration due to the experiment.

Full time of launching the project was 126.283 s, however without circle detection it would be only about 4s. Circle detection is needed only for demonstrating reason so if the speed is necessary user can comment lines for circle detection.

Whole working code is in Project.zip in final_project.m file. We compared values with the theoretical one and values were correct.

7. Code

```
% Paweł Wągrodzki Łukasz Sarnacki


clc
clear
close all
```

```
%% 0. Parameters:
lambda = 0.6328;
M = -10;
dx = 3.45/abs(M);
n0 = 1;
ns = 1.45701;
% 1.Extract the field with TPS formulas -> Lab2

objtype = 'phase';

if objtype =='phase'
    folder_name = 'C:\Users\pawel\OneDrive\Dokumenty\2_stopien\2_semestr\Numerical methods in optics\Projekt_Lab\';
    data1 = double(imread([folder_name,'holo1.bmp']));
    data2 = double(imread([folder_name,'holo2.bmp']));
    data3 = double(imread([folder_name,'holo3.bmp']));
    data4 = double(imread([folder_name,'holo4.bmp']));
    data5 = double(imread([folder_name,'holo5.bmp']));
    ph = atan2(2*(data2-data4),(2*data3-data1-data5));
    amp = (1/4)*sqrt(4*(data2-data4).^2 + (2.*data3-data1-data5).^2);
end

u=amp.*exp(1i*ph); % phase case
u1=u;
uin = u1;

% value calculated from autofocusing
uout = PropagatePlaneWaveDecC2DNxNy(uin,113.7,n0,lambda,dx);
```

```
%% comparing defocused and focused fields

figure
subplot 121
imagesc(abs(uin))
title('Initial amplitude')

subplot 122
f_amp = abs(uout);
imagesc(f_amp)
title('Focused amplitude')


figure
subplot 121
imagesc(ph)
title('Initial phase')

subplot 122

ph_before=angle(uout);
imagesc(ph_before)
title('Focused phase')
colormap gray
```

```
%% unwrapping phase
% function TIE_unwrap2DCdct2

ph_1= angle(uout);
figure
unwrapped_1 = TIE_unwrap2DCdct2(ph_1,dx,lambda);
shape = double(imbinarize(unwrapped_1));
for x=1:1:size(unwrapped_1,1)
    for y=1:1:size(unwrapped_1,2)
        if shape(x,y)==1
            shape(x,y)=unwrapped_1(x,y);
        end
    end
end
```

```
%% Circle detection
figure
subplot 111
colormap gray
imagesc(unwrapped_1)
[centers, radii, metric] = imfindcircles(unwrapped_1,[38 300]);
viscircles(centers, radii,'LineStyle','--');
```

```
%% Analyzed lens
figure
colormap gray
subplot 121
xlabel ('X Pixels')
ylabel('Y Pixels')

unwrapped_2 = unwrapped_1(262:605,363:707);
imagesc(unwrapped_2)
```

```
%% Analyzed part of lens
figure
phase_values = unwrapped_2(164,50:284);
x = 0.35*linspace(0,size(phase_values,2),size(phase_values,2));
plot(x,phase_values)
xlabel('X [um]')
ylabel('Phase [rad]')
title('Height calculation from phase')
h = max (phase_values)*lambda/(2*pi*(ns-1));
```

```
%% Analyze of other lenses
x = 0.35*linspace(0,size(unwrapped_1,2),size(unwrapped_1,2));
y = unwrapped_1(108,1:end);
figure
plot(x,y)
xlabel('X [um]')
ylabel('Phase [rad]')
title('Phase of row of lenses')
```