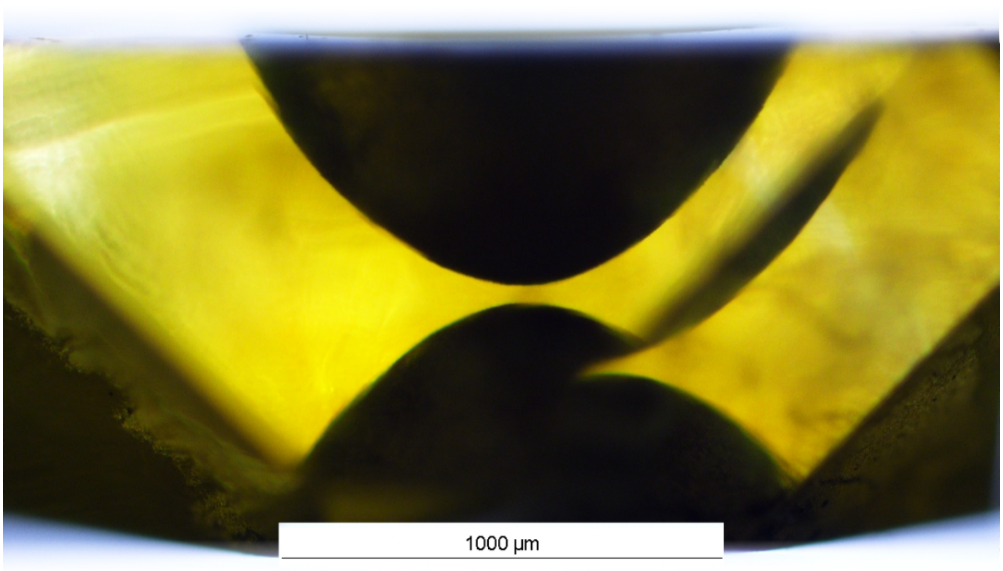
DTU Danchip
National Center for Micro- and Nanofabrication

Laser micromachining of single-crystalline diamond refractive x-ray lens

Author:
Jens Hemmingsen (s135101)



Supervisors:
Kristian Hagsted Rasmussen & Peter Jesper Hanberg

Technical University of Denmark

November 27, 2018

Contents

1 Introduction

X-ray imaging is a widely used non-invasive imaging technique, with applications in fields such as materials science, medicine or even food quality control, where the ability to detect foreign objects can be critical. X-rays have the ability to transmit through many materials with absorption rates dependent on the material density, which is used to create traditional absorption x-ray images.
The ability to transmit through materials is, however, a fundamental problem for any application that requires focusing of the x-rays, since the x-rays will transmit through traditional optical lenses, without any significant refraction.
Focusing of x-rays is currently done by consecutive stacking of beryllium lenses, in what is known as a compound lens, but as the x-ray sources have improved over time, the inherent refractive grain structure in the beryllium lenses have become a source of degradation of the coherence of the beam.

1.1 Project motivation

Lenses made of single-crystalline diamond has the potential to reduce the lens degradation of the x-rays, by drastically reducing the amount of crystal defects present in the lens, as well as having a far better thermal transport ability while still maintaining a relatively low absorption rate [1] [2] [3] [4] [5].

The goal of this project is to design and fabricate, using relatively cheap laser micromachining, a double-concave single-crystalline diamond refractive x-ray lens, as seen in figure 1, for use in a compound x-ray lens, as seen in figure 2.

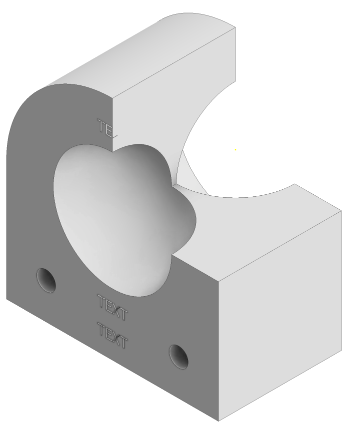


Figure 1: Single unit of a double-concave lens. Each lens unit can only focus the beam slightly, which means the several lenses must be stacked in a compound lens, in order to achieve sufficient focusing.

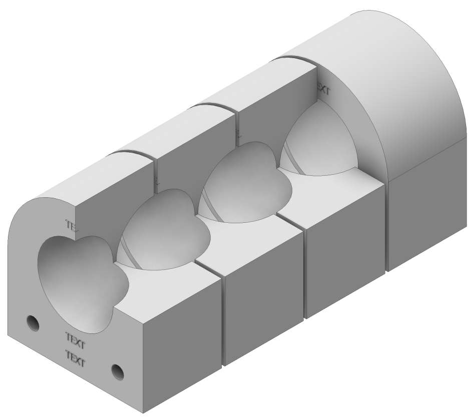


Figure 2: Compound lens constructed from several lenses stacked along the optical axis.

2 Device specifications

The final lens should have a geometry as shown in figure 3, with the following specifications:

- Paraboloid shape.
- RMS roughness of less than 1 µm.
- Radius of curvature R of 150 µm.
- Maximum waist thickness d of 40 µm

2.1 Lens material

The lens is made from single-crystalline HPHT diamond with (100) faces from Sumicrystal, category: UP2512.

2.2 Lens geometry

The complete lens has the shape of a rectangle with a semicircle on top. The lens aperture is centered in the center of the semicircle, and two alignment pinholes are located in the rectangular bottom part of the lens. Various text will be written on the front side of the lens (the upstream side) such as a unique lens ID or aperture dimensions, as seen in figure 3 and 4.

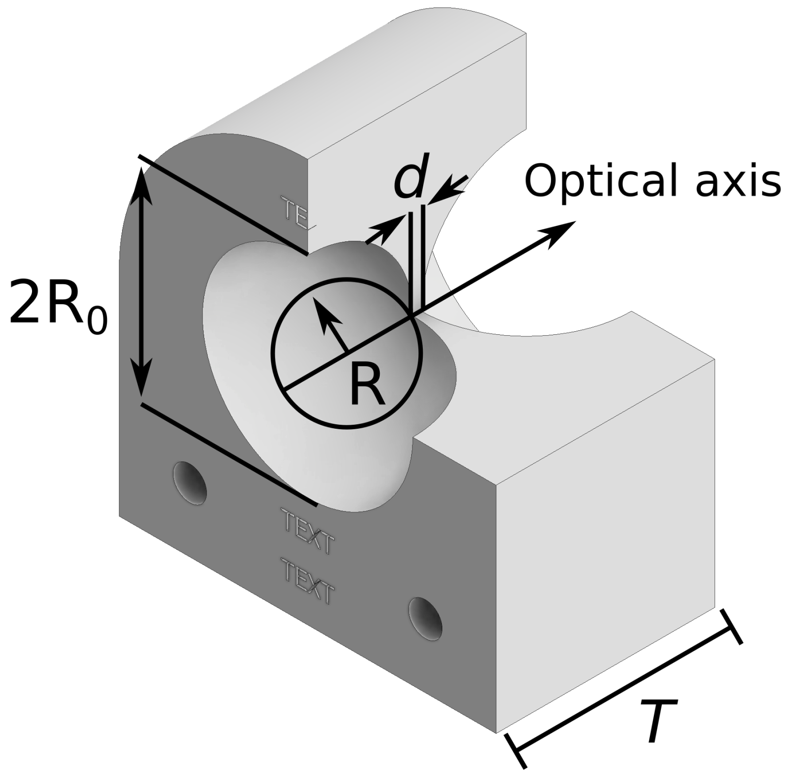


Figure 3: The diameter of the aperture is defined by the thickness of the substrate T and the waist d, as well as the radius of curvature R.

The lens geometry is defined by a set of global fixed dimensions for the overall geometry of the lens and pinholes, a set of specific parameters for the waist thickness d and radius of curvature R of the paraboloids, and finally a calculated aperture diameter $2R_0$. The text on the upstream side is positioned relative to the aperture diameter. A work flow for determining lens parameters might look like this:

1. Measure width, height, and thickness T of the diamond substrate.
2. Decide a waist thickness d, as well as a radius of curvature R for the paraboloid.
3. The aperture diameter $2R_0$ can now be calculated.

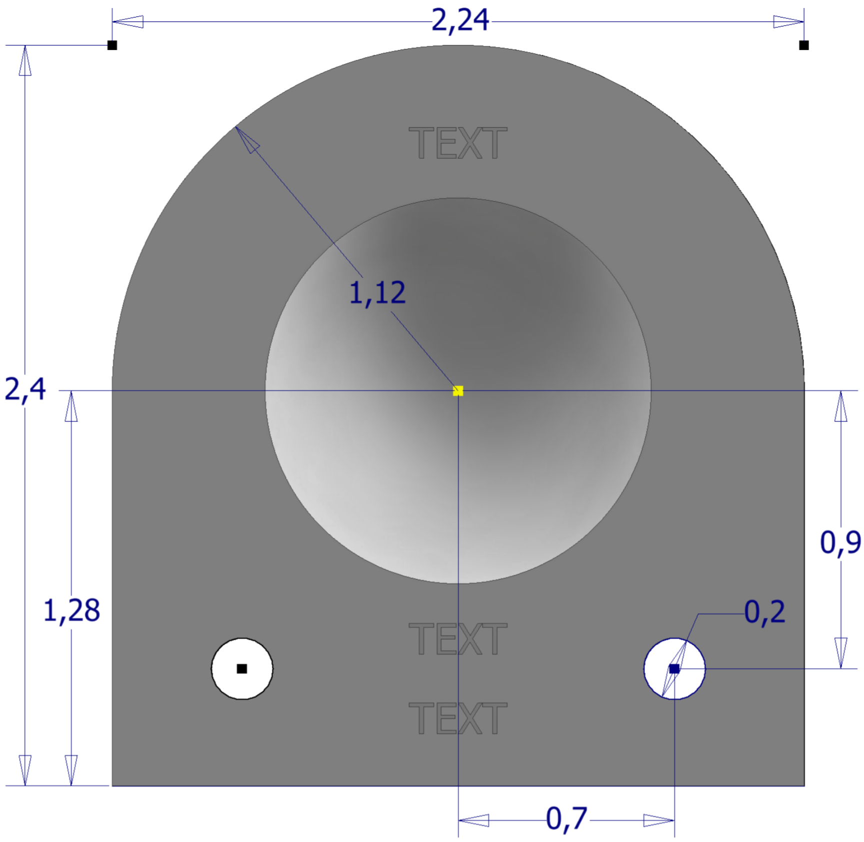


Figure 4: The global dimensions of the lens. The dimensions of the aperture is determined by the thickness of the substrate as well as the desired waist thickness and radius of curvature. All dimensions are in mm.

2.3 Aperture diameter

The aperture diameter $2R_0$ can be calculated by using the following expression with the parameters for substrate thickness T, waist thickness d, and radius of curvature R:

$$2R_0 = 2\sqrt{2R\left(\frac{T}{2} - \frac{d}{2}\right)} \tag{2.1}$$

3 Equipment

3.1 Laser micromachining tool

The laser micromachining tool is a microSTRUCT vario from the company 3D-Micromac AG, with a picosecond, 1064 nm frequency tripled to 355 nm, laser FUEGO from Time Bandwidth.

There are different sets of settings for making the paraboloids and for cutting, shown in table 1. This is due to the requirements of cutting deep narrow trenches, where re-deposition and plasma screening have a much bigger impact on the process compared to the paraboloids, which are mostly open surface processing.

Parameter	**Paraboloid**	**Cutting**
Wavelength:	355 nm	355 nm
Pulse length:	10 ps	10 ps
Repitition rate:	200 kHz	200 kHz
Average power:	0.375 W	3 W
Markspeed:	37.5 mm/s	300 mm/s
Beam diameter (FWHM):	~10 µm	~10 µm
Line overlap:	40%	40%
Ablation rate:	1.23 µm/iteration	0.51 µm/iteration

Table 1: Laser parameters. The settings for making paraboloids and for cutting vary slightly from each other.

3.1.1 Laser ablation rate by line overlap

The line overlap parameter determines how much overlap parallel lines in a line array must have, to have the best milling performance. In this case it was determined that an overlap of 40% gives the lowest roughness. Figure 6 and 7 shows the ablation rate, as function of line overlap percentage, as well as the surface variation accompanied by that line overlap. We decided to go for as high ablation as possible, while keeping the surface variation as low as possible, which meant that we decided to use 40% overlap.

The line overlap is defined as a percentage of the beam diameter, which can be seen in table 1, and an example of two overlapping lines can be seen in figure 5:

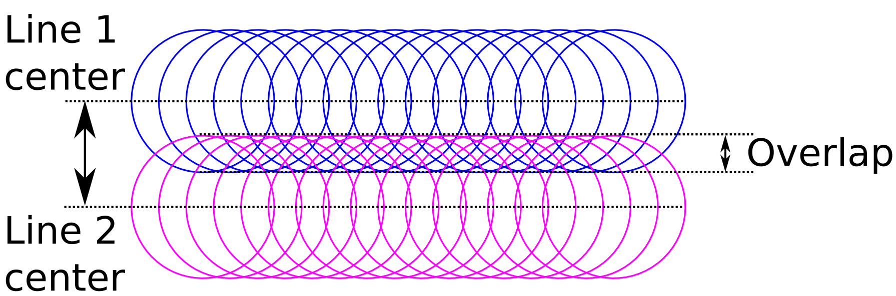


Figure 5: Line overlap of two lines. The overlap is defined as a percentage of the beam diameter, in this project 40% overlap was used.

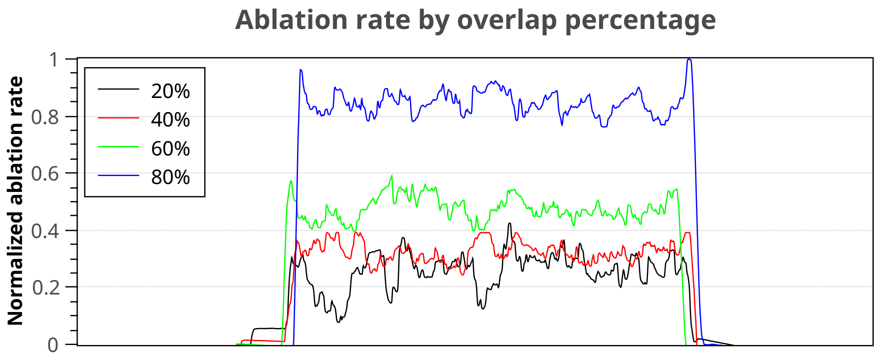


Figure 6: 1D profiles of different 4 overlap percentages. While 80% overlap has a much higher ablation rate than 40%, it also has significant surface variation, while the roughness of the 40% overlap is much lower.

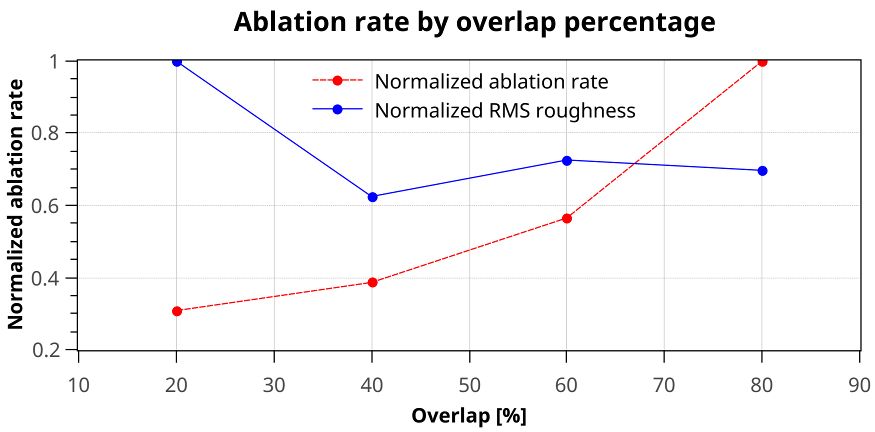


Figure 7: The normalized average ablation rate of each of the 4 overlap percentages. We see from the graph data, that a line overlap of 40% gives the lowest RMS roughness.

3.2 Cleaning equipment

The lenses are cleaned, after laser micromachining, in a heated acidic solution of 1:1:1 of nitric acid, sulphuric acid and perchloric acid. This removes all traces of carbon residues, as well all other dirt left over from the laser processing.

The cleaning process takes at least a full working day; The solution is placed into a tub of sand, heated to boiling point (300°C) for at least 4 hours, and is then left to cool, as seen in figure 8.

The solution should not be able to react with any carbon bound in a diamond lattice, which means that it should not react with the lens, regardless of how long it is left in solution.

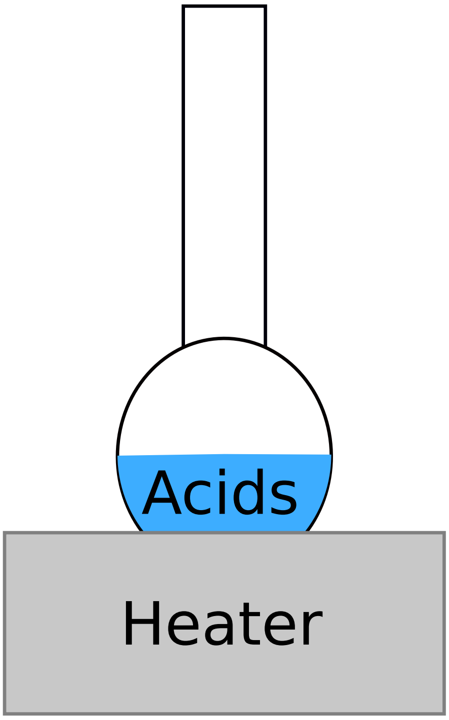


Figure 8: The cleaning setup. The flask containing the diamond and the acidic solution is suspended in a heated tub of sand for at least 4 hours at 300°C.

3.3 Characterization equipment

The lens characterization was done using the following equipment:

Type	**Equipment**
Stylus profilometer:	Dektak XTA
Optical profilometer:	Sensofar PLu Neox 3D
SEM:	Zeiss Supra VP 40

Table 2: Characterization equipment.

4 Laser micromachining

Laser ablation is a method used to remove material from a solid surface, by irradiating it with a laser beam. The material directly underneath the beam is heated and vaporized. The process can be compared to old fashioned machine milling, in which a rotary tool is used to remove material from a solid surface, and indeed laser ablation is also known as laser micromachining. The laser used for this project is working in the hot ablation regime, which is defined by the pulselength. The laser processing is controlled by visual basic scripting, and examples of these can be found in Appendices B, C, and D.

No pre-treatment is necessary before putting the diamond substrates in the laser tool for processing. Should the diamond substrates be very dirty for some reason, a simple cleaning with ethanol and a paper towel should be enough.

It is much more important that the diamond substrates have a flat surface, without any steps or terraces, as any non-uniformity of the surface will translate into the final paraboloid shape.

Normal laser tool usage is not covered in this section, as any operator will need to be trained in the use of the laser tool, before any processing can be done.

The laser tool processing is split into two parts for the milling of the two sides of the diamond. We refer to it as upstream side processing and downstream side processing. After completing the first part, the diamond has to be flipped, so processing can be done on the downstream side.

4.1 Sample loading - both sides

The diamond sample should be loaded into the laser in the "most meaningful way". The samples are not always the same size, but they should be positioned on the laser chuck in such a way, that the rectangular direction of the sample follows the rectangular direction of the lens design, shown in figure 9:

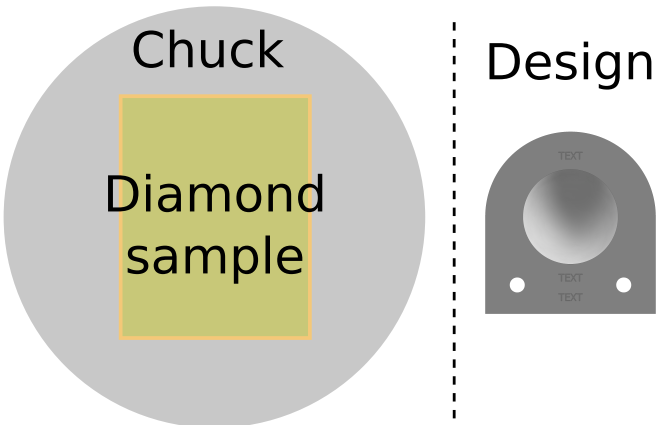


Figure 9: Sample loading on laser chuck viewed next to the design layout.

4.2 Job file preparation - upstream side

1. Measure the thickness T of the diamond substrate with a micrometer ($\pm 1\,\mu m$), before loading it into the laser tool.

2. Open the "upstream side" job file.

3. Edit the "upstream side" visual basic script file. Example script is shown in Appendix B:

- Insert desired radius of curvature: RoC.
- Insert desired waist thickness: d.
- Insert the measured substrate thickness: $lensThickness$.
- Insert the appropriate ID text: $testID\#$.

4. Edit the "cutout upstream side" visual basic script file. Example script is shown in Appendix C:
   - Insert the measured substrate thickness: $lensThickness$.
5. Open the parameter menu and measure the laser power. Note which percentage gives a power of 0.375 W, and which percentage gives a power of 3.0 W.
6. Open the parameter file for the "upstream side" visual basic script.
   - Insert the power percentage for 0.375 W.
7. Open the parameter file for the "cutout upstream side" visual basic script.
   - Insert the power percentage for 3.0 W.

4.3 Alignment - upstream side

1. Move camera to the sample location on the chuck.
2. Add a measurement circle with the same radius as the lens outline, eg. 1.120 mm.
3. Use the measurement circle to set the processing location in a good spot. Remember that the bottom part of the lens design is rectangular.
4. Next align the focus to the top side of the sample.

4.4 Job file preparation - downstream side

1. Open the "downstream side" job file. Example script is shown in Appendix D:
2. Edit the "downstream side" visual basic script file.
   - Insert desired radius of curvature: RoC.
   - Insert desired waist thickness: d.
   - Insert the measured substrate thickness: $lensThickness$.
3. Open the parameter menu and measure the laser power. Note which percentage gives a power of 0.375 W.
4. Open the parameter file for the "downstream side" visual basic script.
   - Insert the power percentage for 0.375 W.

4.5 Alignment - downstream side

1. Move camera to the sample location on the chuck.
2. Adjust the focus below the top side - it is now possible to adjust the x-y location to the already processed paraboloid on the upstream side, as this is visible through the sample.
3. Next align the focus to the top side of the sample.

5 Cleaning

The cleaning is done in a special setup inside a fumehood. The last user should have cleaned the equipment before leaving the setup.

1. Prepare all glasswares.
   - Place an empty glass beaker on the table.
   - Remove the flask from the flask holder, and put it into the glass beaker to hold it.
2. Put samples into the flask.
3. Pour 10 mL of sulphuric acid into the flask.
4. Pour 10 mL of nitric acid into the flask.
5. Pour 10 mL of perchloric acid into the flask.
6. Reattach the flask to the flask holder.
7. Place the flask into the sand bath.
8. Turn on the heater, and set it to 4 hours at boiling point (300°C).

After cleaning is finished, put the samples in a sample holder and proceed to clean all glasswares.

6 Characterization

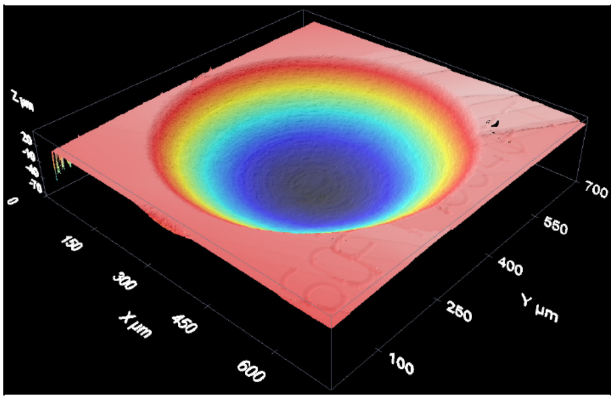
The characterization is primarily done with an optical profiler. A stylus profiler can also be used, but the optical profiler has the benefit of scanning the entire 2D surface at once, while the stylus profiler only makes 1D profiles.

Inspection in SEM is also possible, but since diamond is a very good electrical insulator, great care must be taken to avoid charge build-up in the SEM.

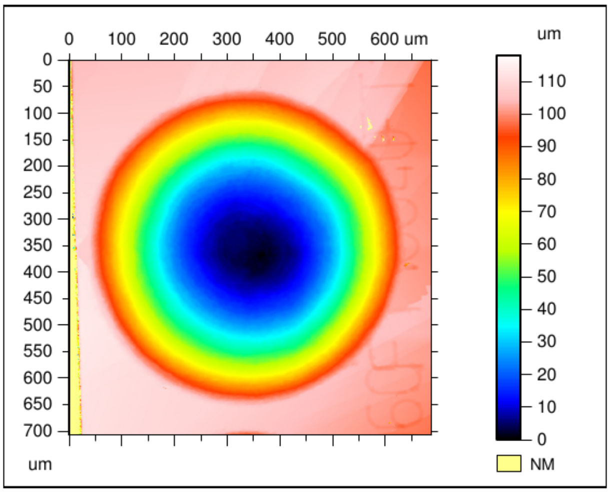
6.1 2D profile measurements

The following images were made from characterization of a lens with an aperture of 600 µm and a radius of curvature of 450 µm, and are meant only as a description of how a measurement in the optical profiler should look; the actual roughness measurements are in the results section.

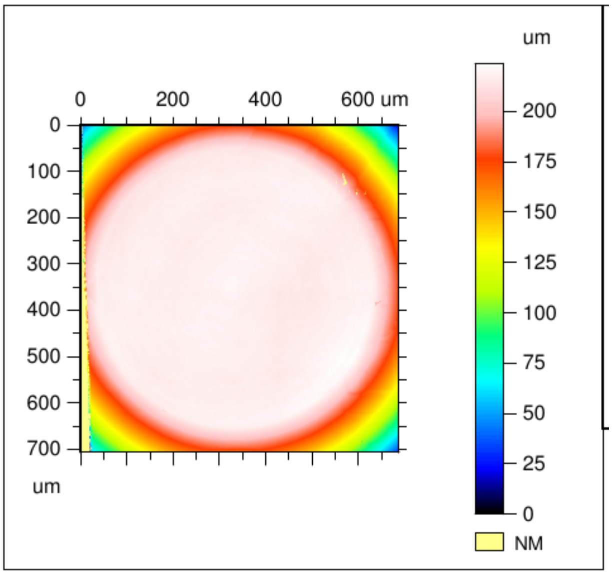
Step 1 - Profile raw data:
The 2D profile data from the optical profiler, represented in isometric 3D view. This image was made in confocal mode, using a 50x lens, stitching 4 images.



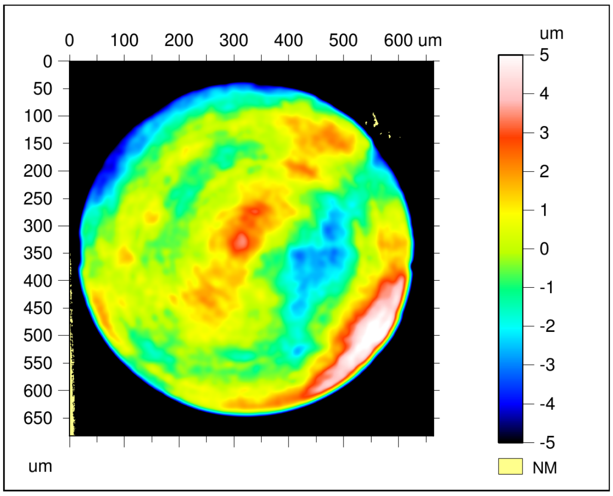
Step 2 - Primary profile:
The primary profile is the extracted surface after levelling.



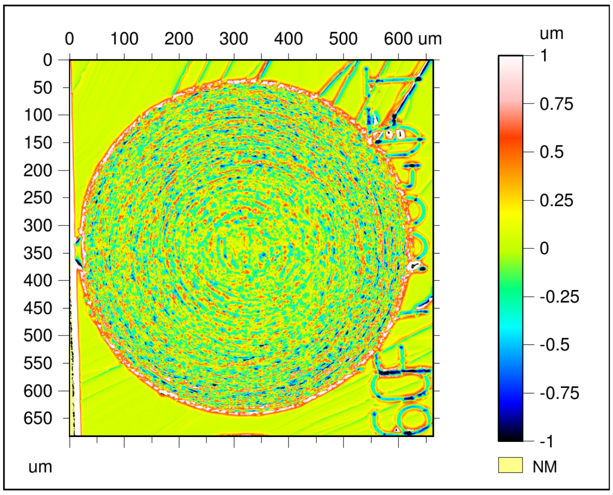
Step 3 - Scale-limited surface:
An ideal 2. order polynomial, fitted to the paraboloid area, is removed. The remaining surface is the scale-limited surface which contains both the roughness and waviness information.



Step 4a - 2D Waviness:
Applying the low-pass filter to the scale-limited surface produces the waviness result.



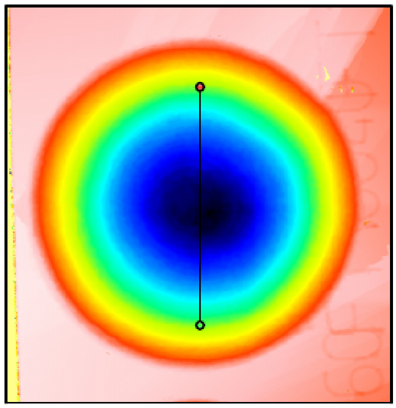
Step 4b - 2D Roughness:
Applying the high-pass filter to the scale-limited surface produces the roughness result.



6.2 1D profile measurements

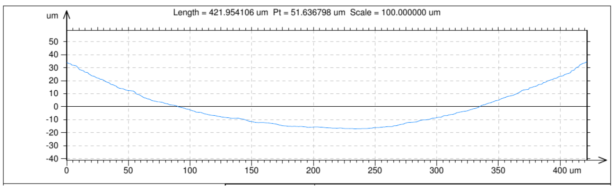
A 1D profile can be made from the 2D data. This data is sometimes easier to interpret, than the 2D data. The following measurements were made from the 2D data from the previous section, of a lens with an aperture of 600 µm and a radius of curvature of 450 µm, and are meant only as a description of how a measurement in the optical profiler should look; the actual roughness measurements are in the results section.

Step 1 - Profile raw data:
The position where the 1D profile is made. The profile length will be indicated on the resulting 1D profile graphs.

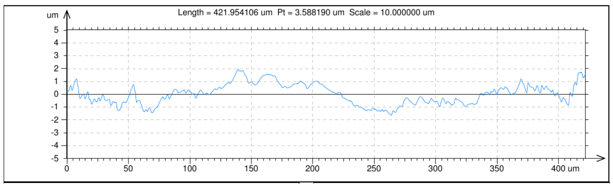


Extracted profile (Topography layer)

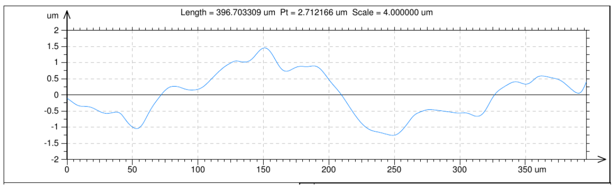
Step 2 - Primary profile:
The primary profile is the extracted profile, from the 2D data, after levelling.



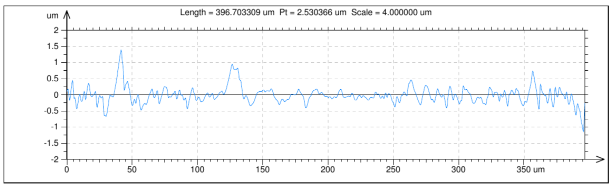
Step 3 - Scale-limited profile:
An ideal 2. order polynomial, fitted to the parabolic profile, is removed. The remaining profile is the scale-limited profile which contains both the roughness and waviness information.



Step 4a - 1D waviness:
Applying the low-pass filter to the scale-limited profile produces the waviness result.



Step 4b - 1D roughness:
Applying the high-pass filter to the scale-limited profile produces the roughness result.



6.3 Profile roughness comparison on three aperture sizes

Due to a limited sample size, no statistics have been done on the roughness, however it was observed that all processed paraboloids have comparable roughness. The following 3 graphs compares the roughness of three different test paraboloids, where it can be seen, that all three paraboloids have comparable roughness. The first image, seen in figure 10, is the roughness profile for a paraboloid with an aperture of 200 µm.

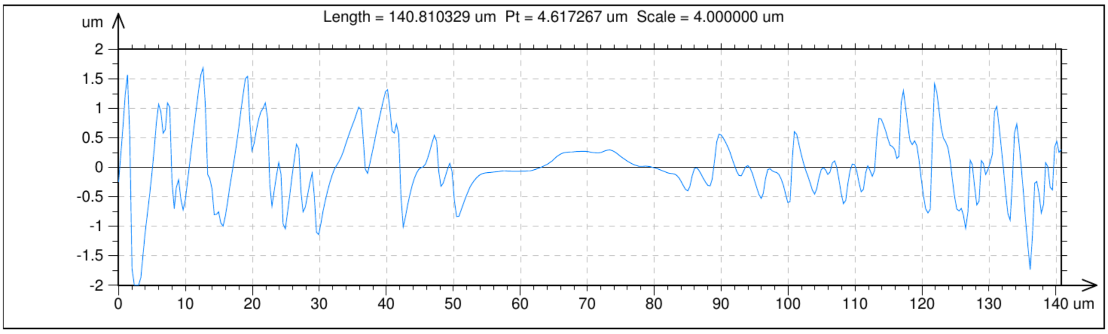


Figure 10: The 1D roughness profile of a paraboloid with an aperture of 200 µm.

The next image, seen in figure 11, is the profile roughness of a paraboloid with an aperture of 400 µm.

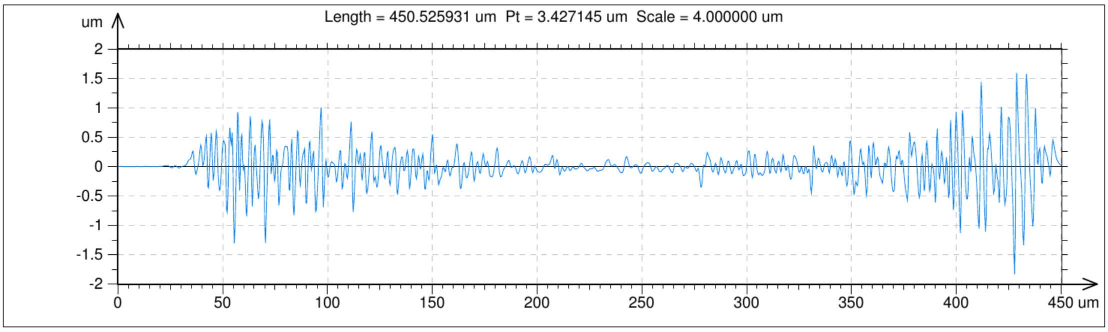


Figure 11: The 1D roughness profile of a paraboloid with an aperture of 400 µm.

The final image, seen in figure 12, is the profile roughness of a paraboloid with an aperture of 850 µm.

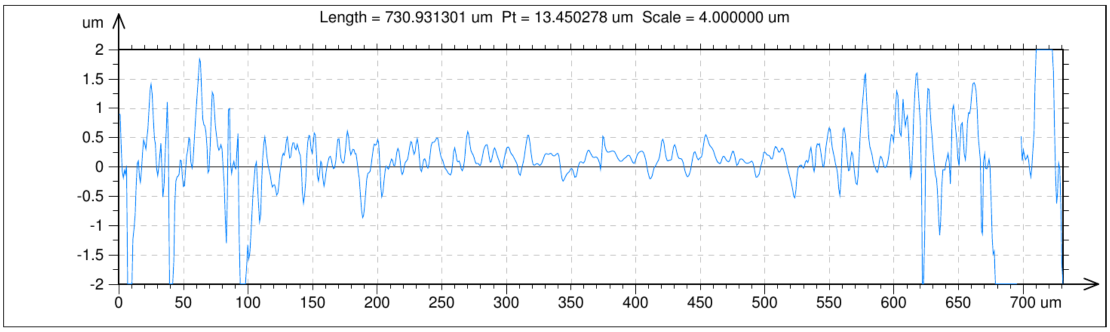


Figure 12: The 1D roughness profile of a paraboloid with an aperture of 850 µm.

6.4 Method for determination of surface roughness and waviness

The surface roughness is defined as the high-frequency changes in the surface, while the waviness is defined as the low-frequency changes.

The method used for determining the surface texture properties is covered in ISO 25178 *Geometric Product Specifications (GPS) – Surface texture: areal*.

The method used for filtering the data to obtain the roughness and waviness data is covered in ISO 16610-31 *Robust profile filters: Gaussian regression filters*.

The application of both of these ISO standards are explained in the paper *Good Practices for the use of areal filters* [6].

The entire process for determining the surface roughness and waviness uses the following method:

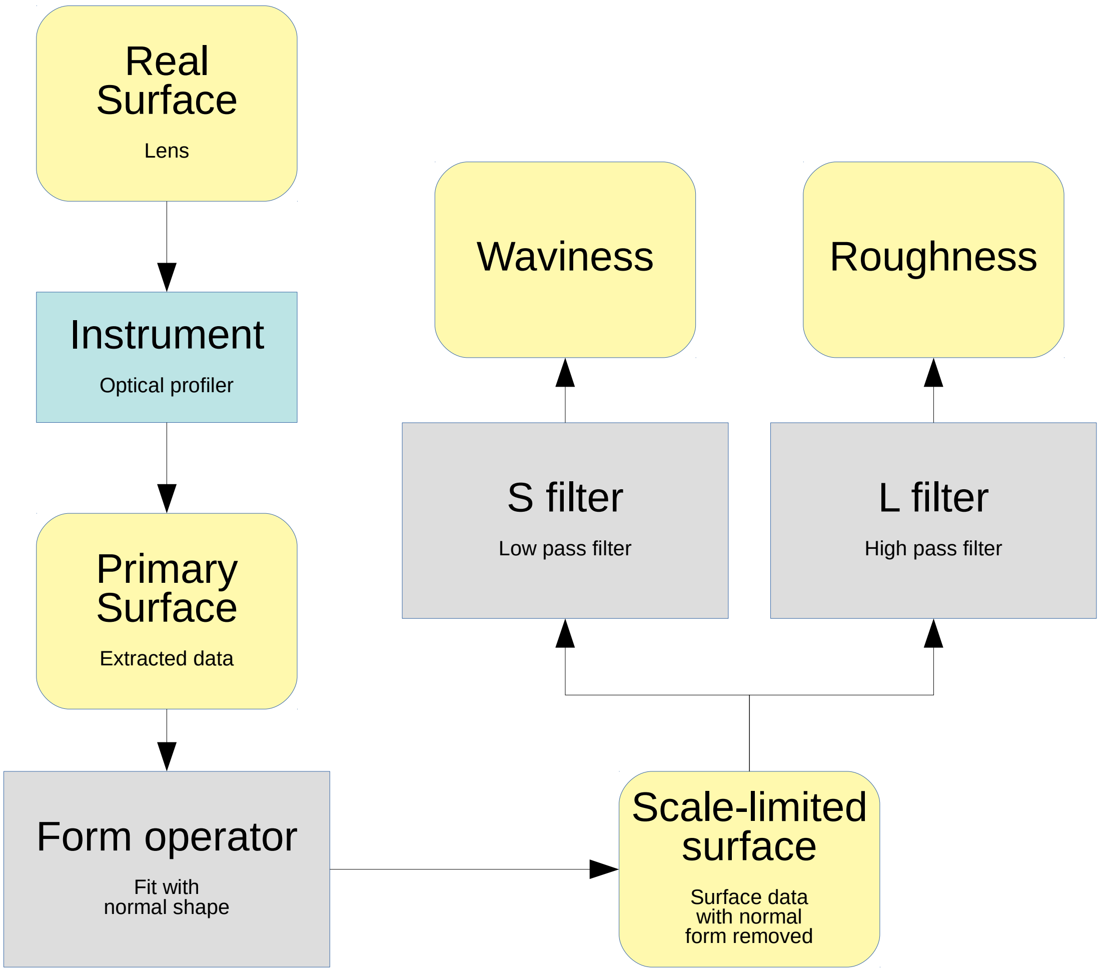


Figure 13: Flow chart for determining surface roughness and waviness.

1. **Real surface:** The sample surface we want to measure.
2. **Instrument:** The instrument used for characterization.
3. **Primary surface:** The data we have collected.
4. **Form operator:** This step fits a 2. order polynomial to the shape.

5. **Scale-limited surface:** This step removes the fitted shape from the primary surface. We are left with a linear combination of the surface roughness and the surface waviness.
6. **S/L filter:** The high- and low-pass filters used to separate the roughness data from the waviness data.
7. **Waviness:** The changes in the surface, which takes place over long distances.
8. **Roughness:** The changes in the surface, which takes place over short distances.

6.5 Filter specifications

The filter, used to separate the roughness data from the waviness data, is a robust gaussian regression filter with a cut-off length of 25 µm.
The filtering is done by making a Fourier transform of the scale-limited surface, and then applying either a high-pass filter to get the roughness (high frequency variations), or a low-pass filter to get the waviness (low frequency variations).
Variations in the surface which happens over distances that are shorter than the filter cut-off length is interpreted as roughness, while variations in the surface which happens over distances that are longer than the filter cut-off length is interpreted as waviness

The exact filter cut-off length is a compromise. If it is set too short, the roughness becomes very large as *any* deviation from an ideal paraboloid shape is counted as roughness. If it is set too large, *no* deviation from an ideal paraboloid is counted as roughness.
This cut-off value of 25 µm was chosen based on the diameter of the beam, which is 10 µm. The cut-off is then 2.5 times the diameter of the beam.
A physical interpretation of this is that any changes in the surface, which takes place over longer distance than 25 µm, is the waviness, while surface-changes shorter than 25 µm is roughness.

Typical values for the cut-off length are 8, 25, or 80 µm. Figure 14 shows a comparison chart of how the cut-off length influences the roughness measurement. Table 3 shows an example of how sensitive the roughness is to the filter cut-off, while table 4 shows that the waviness is not sensitive to the cut-off. This example data is from the final lens on the upstream side.

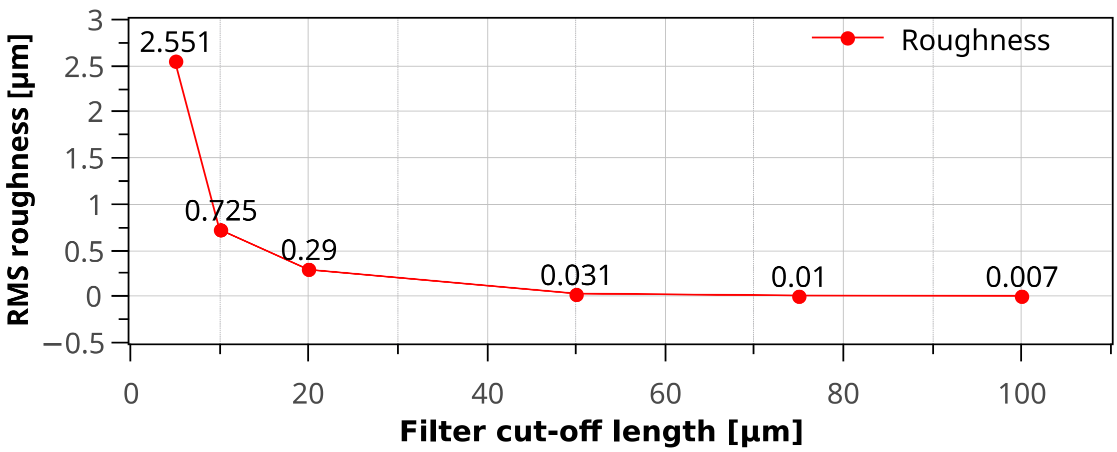


Figure 14: Comparison of RMS roughness as function of the filter cut-off length. If the cut-off is set very low, the roughness increases dramatically, while the opposite is true for very large cut-off lengths.

RMS roughness			
Coverage	8 µm filter	25 µm filter	80 µm filter
[%]	[µm]	[µm]	[µm]
50	1.975	0.130	0.010
75	1.676	0.241	0.011
100	1.282	0.390	0.012

Table 3: Comparison of RMS roughness on the same surface, but with three different filter cut-off lengths, 8, 25, and 80 µm. The roughness value is very sensitive to the cut-off length.

RMS waviness			
Coverage	8 µm filter	25 µm filter	80 µm filter
[%]	[µm]	[µm]	[µm]
50	2.118	2.975	3.006
75	2.214	2.818	2.829
100	2.256	2.536	2.566

Table 4: Comparison of RMS waviness on the same surface, but with three different filter cut-off lengths, 8, 25, and 80 µm. The waviness value is not sensitive to the cut-off length.

6.6 Aperture coverage

The roughness and waviness have been measured in three different aperture coverages; one for the center 50% of the aperture, one for the center 75% of the aperture, and the last one which covers the entire 100% of the aperture.

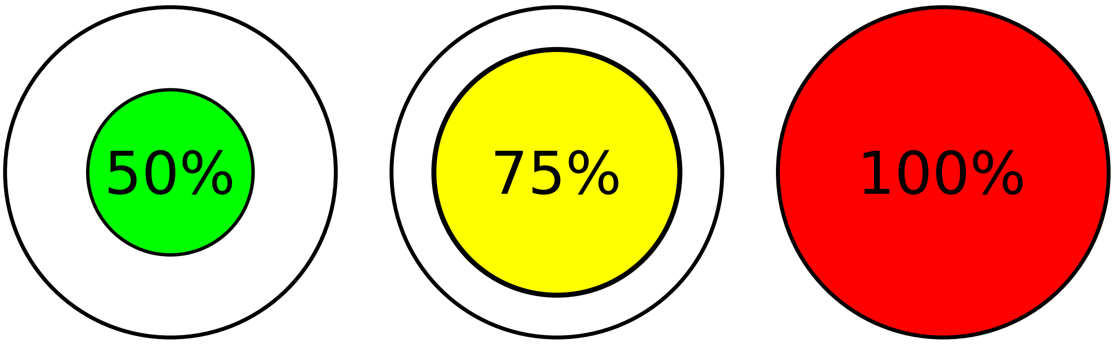


Figure 15: The three different aperture coverages used for representing the roughness and waviness 2D results.

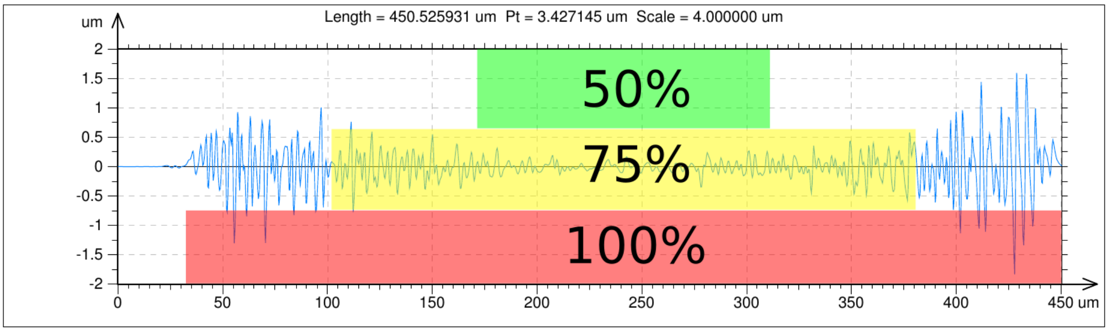


Figure 16: The three different aperture coverages used for representing the roughness and waviness 1D results.

7 Results

7.1 Surface roughness

The following roughness values were measured on the final lens, on the upstream side. Due to the lack of a large number of samples, no statistics have been done comparing the surface roughness over many samples. We did however observe that all the finished paraboloids have comparable roughness.

From table 5 we see that the rms roughness in the inner 50% of the paraboloid is 0.130 µm.

Coverage	**RMS roughness**
[%]	[µm]
50	0.130
75	0.241
100	0.390

Table 5: Typical roughness values for the paraboloids at a filter cut-off of 25 µm. The roughness is the high-pass filtered data from the scale-limited surface.

7.2 Surface waviness

The following waviness values were measured on the final lens, on the upstream side. Due to the lack of a large number of samples, no statistics have been done comparing the surface waviness over many samples. We did however observe that all the finished paraboloids had comparable waviness.

From table 6 we see that the rms waviness in the inner 50% of the paraboloid is 2.975 µm.

Coverage	**RMS waviness**
[%]	[µm]
50	2.975
75	2.818
100	2.536

Table 6: Typical waviness values for the paraboloids at a filter cut-off of 25 µm. The waviness is the low-pass filtered data from the scale-limited surface.

7.3 Radius of curvature

The radius of curvature seems to be a non-issue with this fabrication method. During process optimization, paraboloids were made with a radius of curvature ranging from 50 µm to 500 µm.

7.4 Minimum waist thickness

The smallest waist thickness d measured, on a finished lens, was 23 µm. This was done using the settings specified in table 1.
The waist measurement was done in the optical profiler by first measuring the height of the upstream paraboloid, turning the lens on the other side and measuring the height of the downstream paraboloid. Finally these two height values was subtracted from the total thickness of the substrate, as shown in figure 17:

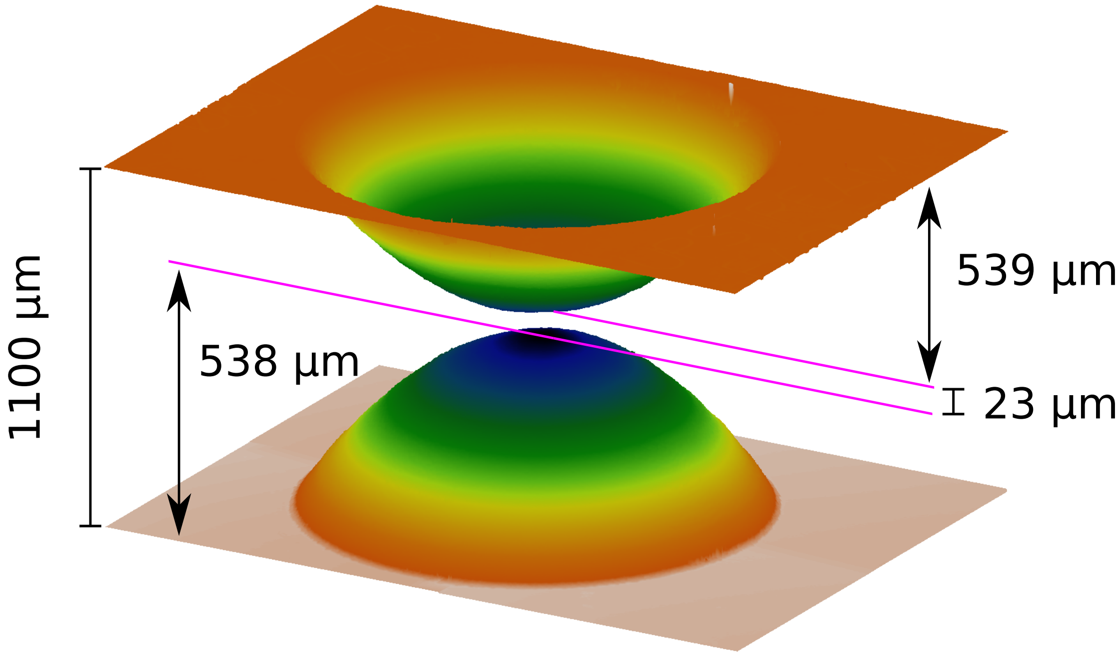


Figure 17: Reconstruction of how the waist was measured in the final lens. Each paraboloid in the lens was measured in the optical profiler and then the sum of the paraboloid heights was subtracted from the substrate thickness to obtain the waist thickness, in this case $d = 23\,\mu m$.

During process optimization, we discovered that the total average beam power has a significant influence on the minimum waist possible; at 3 W average beam power the minimum waist was $\sim 40\,\mu m$, but by reducing the beam power to 0.375 W, it became possible to go to $23\,\mu m$.

Further process optimization could potentially reduce the waist thickness even more.

8 Discussion and conclusion

8.1 Laser processing optimization

The total processing time for a single lens is currently about 60 minutes. This time could probably be reduced in a couple of different ways, such as more advanced control of the ablation, or reducing the amount of features needed for the lens.

The improved laser ablation control could be higher power and faster markspeed, or higher pulse repetition rate, of lower power and slower markspeed, or depth controlled power/speed product.

Reducing the amount of lens features could reduce the processing time, by reducing the total machining length. Relevant features to remove could be the two alignment pinholes, if aligning to the already processed paraboloid is good enough, or maybe the overall shape of the lens could be reduced to a circle, instead of the current shape, which is a semi-circle on top of a rectangle (a Norman window shape).

8.2 Stylus profilometer tip dragging

Stylus profilometers seems to have problems with tip-dragging in parabolic shaped features, altering the parabolic shape, giving it a tilted parabola shape. This was tested on three different stylus profilometers, with various sample holders, and the shape was persistant even when turning the sample 180°, indicating an equipment- or measurement problem. This was further confirmed when measuring the same samples with the optical profiler, where no such tilted parabolic shape was seen.

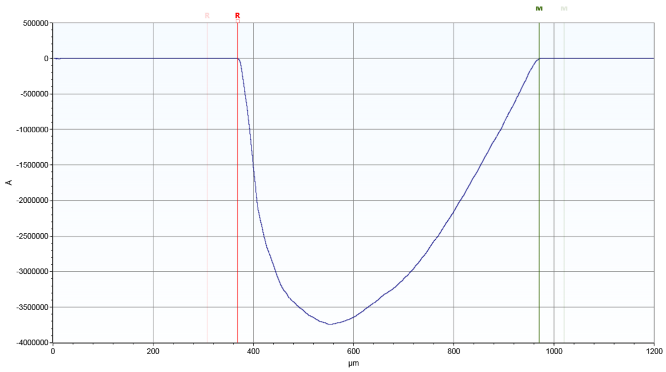


Figure 18: The tilted parabola from a stylus profile measurement.

8.3 SEM charging issues

Diamond is an excellent electrical insulator, which is problematic when trying to use electrons as the imaging medium. Heavy charging effects will influence any SEM image of the cleaned single-crystalline lenses. Tests were made using poly-crystalline diamond instead, which did not exhibit nearly as bad charging. We also observed that processed, but non-cleaned diamonds, had much less charging effects, probably due to the surface carbon dirt acting as a conductive film.

In order to get good quality SEM images of the lenses, we propose to prepare a sample lens for SEM imaging, by coating it with a thin gold film. This can be done using standard metal deposition techniques.

8.4 Conclusion

The goal of the project was reached. We were able to design and fabricate double-concave single-crystalline diamond refractive lenses with surface area roughness, waist thickness, and radius of curvature below the defined limit, using relatively cheap laser micromachining.

The rms roughness of the fabricated lenses was 0.390 µm at 100% aperture coverage, and the waist could be made down to 23 µm. The radius of curvature seems to be a non-issue for this fabrication method.

References

[1] S. Terentyev, M. Polikarpov, I. Snigireva, M. Di Michiel, S. Zholudev, V. Yunkin, S. Kuznetsov, V. Blank, and A. Snigirev, "Linear parabolic single-crystal diamond refractive lenses for synchrotron x-ray sources," *Journal of Synchrotron Radiation*, vol. 24, 01 2017.

[2] V. S. I. S. A. Polikarpov, Maxim Polikarpov, "Diamond x-ray refractive lenses with high acceptance," *Physics Procedia*, 12 2016.

[3] S. Terentyev, V. Blank, S. Polyakov, S. Zholudev, A. Snigirev, M. Polikarpov, T. Kolodziej, J. Qian, H. Zhou, and Y. Shvyd'ko, "Parabolic single-crystal diamond lenses for coherent x-ray imaging," *Applied Physics Letters*, vol. 107, 9 2015.

[4] S. Antipov, S. Baryshev, J. E. Butler, O. Antipova, Z. Liu, and S. Stoupin, "Single-crystal diamond refractive lens for focusing x-rays in two dimensions," *Journal of Synchrotron Radiation*, vol. 23, 1 2016.

[5] S. Antipov, S. Baryshev, S. Baturin, R. Kostin, O. Antipova, and T. Irving, "Single crystal diamond x-ray lens development," 05 2016.

[6] F. Blateyron, "*Good practices for the use of areal filters*," 05 2014.

A Data file references

The data for ablation rate as function of line overlap can be found in the file "chararcterization files/stylus profiler/line density test/new line density graph.ods"

B Example VB script: Upstream side

```
'***************************************************
'
' Diamond laser micromachining
' Test pattern:
'   Paraboloid with line array
'   Rotates [360/total iterations] degrees for each iteration
'   Starts at edge, then moves towards center
'   Focus adjust every 100 um
'
'***************************************************
'
' Declaration
'
'***************************************************

Option Explicit
dim beamDiam, linePitch, lineNums, aperture, iterations, lensHeight, lensWidth,
    lensThickness, oldDiam, deltaX, deltaY, RoC,smallD 'experiment variables (
    ablation rate, sample size, pinholes)
dim radius, theta, iterRotation, penColor, xValue, coef, ablationRate, depth, a,
    sqrCoef 'experiment variables (rotated line array)
dim focusHome, focusAdjust, currentFocus, currentIter, focusAxis 'experiment
    variables (focus adjust)
dim testID1, testID2, testID3, testID4, lineOverlap, oldMarkSpeed 'experiment
    variables (text writer)

dim i,j,k,m,n 'multi-use variables
dim fnt, str, labelSize, oldLabelSize 'printing
dim pi 'calculate pi

loadInitialization ' loads initialization of the VB environment



'***************************************************
'
' script
'
'***************************************************

'*** Test parameters ***
' RoC:                  radius of curvature
' testID:               ID number that will be printed for this test
' aperture:             diameter of the paraboloid opening, [mm]
' depth:                the total depth of the paraboloid (from the diamond
    surface) [mm]

' beamDiam:             diameter of the laser beam, [mm]
' lineOverlap:          how much consecutive lines will overlap, as a percentage
    of the beam diameter, [%]
' ablationRate:         ablation rate per iteration [mm]
' focusAdjust:          the threshold for adjusting the focus. Do not make too
    small, as the focus is adjusted mechanically, which takes a long time


RoC = 0.150
smallD = 0.010
```

```
lensThickness = 1.174

depth = lensThickness / 2 - smallD / 2'0.600
aperture = 2 * Sqr(2 * depth * RoC)

testID1 = "A" & Round(aperture*1000) & " Z" & Round(depth*1000)
testID2 = "UPSTREAM"
testID3 = "d=10"
testID4 = "SERIAL"

beamDiam = 0.010
lineOverlap = 0.40
ablationRate = 0.00123 ' rate for 0.375W @ 37.5 mm/s
focusAdjust = 0.100
lensHeight = 2.400
lensWidth = 2.240


'*** Execute function(s) ***
focusAxis = "ZA"
focusHome = ZA.Pos 'ZA = lens axis, ZB = camera axis

'pinholeAlignMarks
paraboloid
textWriter1
textWriter2
textWriter3
textWriter4



'*****************************************************
'
' subs
'
'*****************************************************

sub pinholeAlignMarks
    iterations = 30'Round(lensThickness/ablationRate)
    echo "Pinhole iterations: " & iterations
    oldDiam = aperture
    aperture = 0.050 'pinhole diameter

    deltaX = -lensWidth/2 + lensWidth*0.1 + aperture/2
    deltaY = -lensHeight/2 + lensHeight*0.1 + aperture/2
    Translate deltaX, deltaY
    circleLineArray

    deltaX = -2*(-lensWidth/2 + lensWidth*0.1)
    deltaY = 0
    Translate deltaX, deltaY
    circleLineArray

    deltaX = -lensWidth/2 + lensWidth*0.1
    deltaY = -1*(-lensHeight/2 + lensHeight*0.1)
    Translate deltaX, deltaY

    aperture = oldDiam
end sub

'*** circular line array ***
sub circleLineArray
linePitch = beamDiam * (1-lineOverlap) ' calculate the distance between
    neighboring lines

```

```
for k = 0 to iterations-1
    penColor = (k/iterations) * 255 ' only relevant for laser GUI
    PreviewPenColor RGB(255, penColor, 0) ' only relevant for laser GUI

    iterRotation = k * (360/iterations) * (pi/180) ' calculate the rotation from the total number of iterations and the current iteration
    radius = (aperture/2) ' calculate the current radius of the circle
    lineNums = aperture / linePitch ' calculate the total number of lines for current iteration

    ' lines
    for j = 0 to lineNums-1
        xValue = (-radius + linePitch/2 + j*linePitch) ' calculate the x-value for the current line
        coef = xValue/radius ' coefficient for use with arctan function

        if (xValue = -1) or (xValue = 1) then
            theta = pi/2 ' "manually" calculate theta = arctan(infinity) = pi/2
        else
            theta = Atn(-coef/Sqr(-coef*coef+1)) + 2*Atn(1) ' else automatically calculate theta
        end if

        n = k mod 2 ' determine even or odd iteration number rotate additional +-90 (or +-45 for 4 iterations, which would otherwise only be +-90)

        if (n = 0) and (iterations <> 4) then
            ' if even AND iterations not equal to 4:
            JumpPol radius, theta + iterRotation
            MarkPol radius, -theta + iterRotation
        elseif (n = 0) and (iterations = 4) then
            ' if even AND iterations equal to 4:
            JumpPol radius, theta + iterRotation + pi/4
            MarkPol radius, -theta + iterRotation + pi/4
        else
            ' if odd:
            JumpPol radius, theta + iterRotation + pi/2
            MarkPol radius, -theta + iterRotation + pi/2
        end if
    next
next
end sub



'*** Paraboloid using rotating line arrays ***
sub paraboloid
    linePitch = beamDiam * (1-lineOverlap) ' calculate the distance between neighboring lines

    ' calculate iterations for paraboloid
    a = depth / (aperture/2)^2 ' calculate a-constant
    iterations = depth/ablationRate
    echo "Total paraboloid iterations: " & Round(iterations)

    str = "Setting relative focus height to: "
    returnFocusHome(str)
    for k = 0 to iterations-1
        penColor = (k/iterations) * 255 ' only relevant for laser GUI
        PreviewPenColor RGB(255, penColor, 0) ' only relevant for laser GUI

        iterRotation = k * (360/iterations) * (pi/180) ' calculate the rotation from the total number of iterations and the current iteration

        sqrCoef = (depth - k*ablationRate) / a
        if (sqrCoef > 0) then
            radius = Sqr(sqrCoef) ' calculate the current radius of the circle
```

```
        else
            radius = 0
        end if

        lineNums = (radius * 2) / linePitch ' calculate the total number of lines for current iteration

        ' line array
        for j = 0 to lineNums-1
            xValue = (-radius + linePitch/2 + j*linePitch) ' calculate the x-value for the current line
            coef = xValue/radius ' coefficient for use with arctan function

            if (xValue = -1) or (xValue = 1) then
                theta = pi/2 ' "manually" calculate theta = arctan(infinity) = pi/2
            else
                theta = Atn(-coef/Sqr(-coef*coef+1)) + 2*Atn(1) ' else automatically calculate theta
            end if

            n = k mod 2 ' determine even or odd iteration number rotate additional +-90 (or +-45 for 4 iterations, which would otherwise only be +-90)

            if (n = 0) and (iterations <> 4) then
                ' if even AND iterations not equal to 4:
                JumpPol radius, theta + iterRotation
                MarkPol radius, -theta + iterRotation
            elseif (n = 0) and (iterations = 4) then
                ' if even AND iterations equal to 4:
                JumpPol radius, theta + iterRotation + pi/4
                MarkPol radius, -theta + iterRotation + pi/4
            else
                ' if odd:
                JumpPol radius, theta + iterRotation + pi/2
                MarkPol radius, -theta + iterRotation + pi/2
            end if
        next
        Call translateFocus(k,str)
    next
end sub

'*** text writer ***
sub textWriter1
    returnFocusHome(str)
    oldMarkSpeed = MarkSpeed ' save testing MarkSpeed
    Markspeed = 100 ' change MarkSpeed to text-default speed
    Jump 0, (aperture/2) + 0.05
    for i = 1 to 20
        fnt.Align = alignCenter
        printer(testID1) ' print test ID
    next
    MarkSpeed = oldMarkSpeed ' reload testing MarkSpeed
end sub

'*** text writer ***
sub textWriter2
    oldMarkSpeed = MarkSpeed ' save testing MarkSpeed
    Markspeed = 100 ' change MarkSpeed to text-default speed
    Jump 0, -(aperture/2) - 0.05 - labelsize
    for i = 1 to 20
        fnt.Align = alignCenter
        printer(testID2)
    next
```

```
    MarkSpeed = oldMarkSpeed ' reload testing MarkSpeed
end sub


'*** text writer ***
sub textWriter3
    oldMarkSpeed = MarkSpeed ' save testing MarkSpeed
    Markspeed = 100 ' change MarkSpeed to text-default speed
    Jump 0, -(aperture/2) - 2*0.05 - 2*labelsize
    for i = 1 to 20
        fnt.Align = alignCenter
        printer(testID3)
    next
    MarkSpeed = oldMarkSpeed ' reload testing MarkSpeed
end sub


'*** text writer ***
sub textWriter4
    oldMarkSpeed = MarkSpeed ' save testing MarkSpeed
    Markspeed = 100 ' change MarkSpeed to text-default speed
    Jump 0, -(aperture/2) - 3*0.05 - 3*labelsize
    for i = 1 to 20
        fnt.Align = alignCenter
        printer(testID4)
    next
    MarkSpeed = oldMarkSpeed ' reload testing MarkSpeed
end sub


'*** translate focus depth ***
sub translateFocus(currentIter, str)
    if ((currentIter*ablationRate + currentFocus) > focusAdjust) then 'if current
     z value is greater than focusAdjust value, then adjust focus
        MoveRel ZA, -(currentIter*ablationRate + currentFocus)
        currentFocus = -currentIter*ablationRate ' save new current focus height
        echo str & currentFocus
        WaitUntilInPos ZA
        Wait 100
    end if
end sub



'*** return to focus home ***
sub returnFocusHome(str)
    currentFocus = 0
    echo str & currentFocus
    Move ZA, focusHome
    WaitUntilInPos ZA
    Wait 100
end sub



'*** Print text  ***
sub printer(str)
    PreviewPenColor RGB(255, 255, 0) ' only relevant for laser GUI
    fnt.Height = labelSize
  print fnt, str
end sub



'*** VB initialization ***
sub loadInitialization
    'Preview settings (pensize and color)
```

```
    previewpensize 0.001
    PreviewPenColor RGB(0, 255, 0)

    ' font type and size
    Set fnt = CreateFont("din1451h")
    labelSize = 0.1 'mm

    ' calculate pi
    pi = 4 * Atn(1)
end sub
```

C Example VB script: Cutout upstream side

```
'*****************************************************
'
' Diamond laser micromachining
' Test pattern:
'   Paraboloid with line array
'   Rotates [360/total iterations] degrees for each iteration
'   Starts at edge, then moves towards center
'   Focus adjust every 100 um
'
'*****************************************************
'
' Declaration
'
'*****************************************************

Option Explicit
dim beamDiam, linePitch, lineoverlap, lineNums, aperture, iterations, lensHeight,
     lensWidth, lensThickness, deltaX, deltaY 'experiment variables (ablation
    rate, sample size, pinholes)
dim radius, theta, iterRotation, penColor, xValue, coef, ablationRate, depth, a,
    sqrCoef 'experiment variables (rotated line array)
dim focusHome, focusAdjust, currentFocus, currentIter, focusAxis 'experiment
    variables (focus adjust)
dim cutoutIterations, xOff, yOff, overlap, numLines 'experiment variables (cut-
    out)

dim i,j,k,m,n 'multi-use variables
dim fnt, str, labelSize, oldLabelSize 'printing
dim pi 'calculate pi

loadInitialization ' loads initialization of the VB environment



'*****************************************************
'
' script
'
'*****************************************************

'*** Test parameters ***
' testID:                ID number that will be printed for this test
' aperture:              diameter of the paraboloid opening, [mm]
' depth:                 the total depth of the paraboloid (from the diamond
    surface) [mm]

' beamDiam:              diameter of the laser beam, [mm]
' lineOverlap:           how much consecutive lines will overlap, as a percentage
    of the beam diameter, [%]
```

```
' ablationRate:          ablation rate per iteration [mm]. (80%, 60%, 40%, 20%) = (0.00240, 0.00163, 0.00131, 0.00117)
' focusAdjust:           the threshold for adjusting the focus. Do not make too small, as the focus is adjusted mechanically, which takes a long time

' pinholeIterations:     number of iterations for pinholes
' cutooutIterations:     number of iterations for cut-out

lensThickness = 1.300

beamDiam = 0.010
lineOverlap = 0.40
ablationRate = 0.000505 ' lines ablation rate = 0.505 um/iteration. Ablation rate for lines at high speed and high power
focusAdjust = 0.100
lensHeight = 2.400
lensWidth = 2.240

cutoutIterations = lensThickness / ablationRate


'*** Execute function(s) ***
focusAxis = "ZA"
focusHome = ZA.Pos 'ZA = lens axis, ZB = camera axis

cutOut
pinholeAlignMarks

'****************************************************
'
' subs
'
'****************************************************


sub pinholeAlignMarks
    iterations = lensThickness/ablationRate
    echo "Total pinhole iterations: 2x " & Round(iterations)
    aperture = 0.300 'pinhole diameter

    deltaX = -lensWidth/2 + lensWidth*0.1 + aperture/2
    deltaY = -lensHeight/2 + lensHeight*0.1 + aperture/2
    Translate deltaX, deltaY
    circleLineArray

    deltaX = -2*(-lensWidth/2 + lensWidth*0.1 + aperture/2)
    deltaY = 0
    Translate deltaX, deltaY
    circleLineArray

    deltaX = -lensWidth/2 + lensWidth*0.1
    deltaY = -1*(-lensHeight/2 + lensHeight*0.1)
    Translate deltaX, deltaY

end sub

'*** circular line array ***
sub circleLineArray
linePitch = beamDiam * (1-lineOverlap) ' calculate the distance between neighboring lines

returnFocusHome(str)

for k = 0 to iterations-1
    penColor = (k/iterations) * 255 ' only relevant for laser GUI
    PreviewPenColor RGB(255, penColor, 0) ' only relevant for laser GUI
```

```
    iterRotation = k * (360/iterations) * (pi/180) ' calculate the rotation from the total number of iterations and the current iteration
    radius = (aperture/2) ' calculate the current radius of the circle
    lineNums = aperture / linePitch ' calculate the total number of lines for current iteration

    ' lines
    for j = 0 to lineNums-1
        xValue = (-radius + linePitch/2 + j*linePitch) ' calculate the x-value for the current line
        coef = xValue/radius ' coefficient for use with arctan function

        if (xValue = -1) or (xValue = 1) then
            theta = pi/2 ' "manually" calculate theta = arctan(infinity) = pi/2
        else
            theta = Atn(-coef/Sqr(-coef*coef+1)) + 2*Atn(1) ' else automatically calculate theta
        end if

        n = k mod 2 ' determine even or odd iteration number rotate additional +-90 (or +-45 for 4 iterations, which would otherwise only be +-90)

        if (n = 0) and (iterations <> 4) then
            ' if even AND iterations not equal to 4:
            JumpPol radius, theta + iterRotation
            MarkPol radius, -theta + iterRotation
        elseif (n = 0) and (iterations = 4) then
            ' if even AND iterations equal to 4:
            JumpPol radius, theta + iterRotation + pi/4
            MarkPol radius, -theta + iterRotation + pi/4
        else
            ' if odd:
            JumpPol radius, theta + iterRotation + pi/2
            MarkPol radius, -theta + iterRotation + pi/2
        end if
    next
    Call translateFocus(k,str)
next
end sub


'*** lens cut out ***
' Lens cut-out dimensions are based on crystal dimensions (width, height) * 0.8.
' The 20% loss is arbitrarily chosen to have 10% cutting loss on every side,
' to minimize any edge flaws in the raw crystal
sub cutOut
    PreviewPenColor RGB(0, 0, 255) ' only relevant for laser GUI

    echo "Total cut-out iterations: " & Round(cutoutIterations)

    str = "Setting relative focus height to: "
    returnFocusHome(str)

    for j = 0 to cutoutIterations
        overlap = 0.2 'line overlap
        numLines = 8
        for i = 0 to numLines - 1
            xOff = i * beamDiam * (1 - overlap) 'x offset per iteration
            yOff = i * beamDiam * (1 - overlap) 'y offset per iteration

            Jump -(lensWidth/2 + xOff), -(lensHeight/2 + yOff)

            MarkRel lensWidth + 2*xOff, 0
            MarkRel 0, lensHeight/2 - yOff
```

```
                ArcRel -(lensWidth/2 + xOff), 0 + 2*yOff, TRUE, -(lensWidth + 2*xOff), 0 - yOff ' ArcRel: [arc RELATIVE TO CURRENT POS center x and y] , [CCW] , [arc end point x and y]
                MarkRel 0, -(lensHeight/2 - 2*yOff)
            next
            Call translateFocus(j,str)
        next
end sub

'*** translate focus depth ***
sub translateFocus(currentIter, str)
        if ((currentIter*ablationRate + currentFocus) > focusAdjust) then 'if current z value is greater than focusAdjust value, then adjust focus
            MoveRel ZA, -(currentIter*ablationRate + currentFocus)
            currentFocus = -currentIter*ablationRate ' save new current focus height
            echo str & currentFocus
            WaitUntilInPos ZA
            Wait 100
            echo "New ZA.pos: " & ZA.pos
        end if
end sub


'*** return to focus home ***
sub returnFocusHome(str)
        currentFocus = 0
        echo str & currentFocus
        Move ZA, focusHome
        WaitUntilInPos ZA
        Wait 100
        echo "New ZA.pos: " & ZA.pos
end sub


'*** Print text ***
sub printer(str)
        PreviewPenColor RGB(255, 255, 0) ' only relevant for laser GUI
        fnt.Height = labelSize
    print fnt, str
end sub


'*** VB initialization ***
sub loadInitialization
        'Preview settings (pensize and color)
        previewpensize 0.001
        PreviewPenColor RGB(0, 255, 0)

        ' font type and size
        Set fnt = CreateFont("din1451h")
        labelSize = 0.1 'mm

        ' calculate pi
        pi = 4 * Atn(1)
end sub
```

D Example VB script: Downstream side

```
'*****************************************************
'
' Diamond laser micromachining
' Test pattern:
'   Paraboloid with line array
'   Rotates [360/total iterations] degrees for each iteration
'   Starts at edge, then moves towards center
'   Focus adjust every 100 um
'
'*****************************************************
'
' Declaration
'
'*****************************************************

Option Explicit
dim beamDiam, linePitch, lineoverlap, lineNums, aperture, iterations, lensHeight,
     lensWidth, lensThickness, RoC, smallD 'experiment variables (ablation rate,
    sample size, pinholes)
dim radius, theta, iterRotation, penColor, xValue, coef, ablationRate, depth, a,
    sqrCoef 'experiment variables (rotated line array)
dim focusHome, focusAdjust, currentFocus, currentIter, focusAxis 'experiment
    variables (focus adjust)

dim i,j,k,m,n 'multi-use variables
dim fnt, str, labelSize, oldLabelSize 'printing
dim pi 'calculate pi

loadInitialization ' loads initialization of the VB environment



'*****************************************************
'
' script
'
'*****************************************************

'*** Test parameters ***
' RoC:                    radius of curvature
' testID:                 ID number that will be printed for this test
' aperture:               diameter of the paraboloid opening, [mm]
' depth:                  the total depth of the paraboloid (from the diamond
    surface) [mm]

' beamDiam:               diameter of the laser beam, [mm]
' lineOverlap:            how much consecutive lines will overlap, as a percentage
    of the beam diameter, [%]
' ablationRate:           ablation rate per iteration [mm]. (80%, 60%, 40%, 20%) =
    (0.00240, 0.00163, 0.00131, 0.00117)
' focusAdjust:            the threshold for adjusting the focus. Do not make too
    small, as the focus is adjusted mechanically, which takes a long time



RoC = 0.150
smallD = 0.010
lensThickness = 1.174

depth = lensThickness / 2 - smallD / 2'0.600
aperture = 2 * Sqr(2 * depth * RoC)

beamDiam = 0.010
lineOverlap = 0.40
ablationRate = 0.00123 ' rate for 0.375W @ 37.5 mm/s
focusAdjust = 0.100
lensHeight = 2.40
```

```
lensWidth = 2.24

'*** Execute function(s) ***
focusAxis = "ZA"
focusHome = ZA.Pos 'ZA = lens axis, ZB = camera axis

paraboloid

'***************************************************
'
' subs
'
'***************************************************



'*** Paraboloid using rotating line arrays ***
sub paraboloid
    linePitch = beamDiam * (1-lineOverlap) ' calculate the distance between neighboring lines

    ' calculate iterations for paraboloid
    a = depth / (aperture/2)^2 ' calculate a-constant
    iterations = depth/ablationRate
    echo "Total paraboloid iterations: " & Round(iterations)

    str = "Setting relative focus height to: "
    returnFocusHome(str)
    for k = 0 to iterations-1
        penColor = (k/iterations) * 255 ' only relevant for laser GUI
        PreviewPenColor RGB(255, penColor, 0) ' only relevant for laser GUI

        iterRotation = k * (360/iterations) * (pi/180) ' calculate the rotation from the total number of iterations and the current iteration

        sqrCoef = (depth - k*ablationRate) / a
        if (sqrCoef > 0) then
            radius = Sqr(sqrCoef) ' calculate the current radius of the circle
        else
            radius = 0
        end if

        lineNums = (radius * 2) / linePitch ' calculate the total number of lines for current iteration

        ' line array
        for j = 0 to lineNums-1
            xValue = (-radius + linePitch/2 + j*linePitch) ' calculate the x-value for the current line
            coef = xValue/radius ' coefficient for use with arctan function

            if (xValue = -1) or (xValue = 1) then
                theta = pi/2 ' "manually" calculate theta = arctan(infinity) = pi/2
            else
                theta = Atn(-coef/Sqr(-coef*coef+1)) + 2*Atn(1) ' else automatically calculate theta
            end if

            n = k mod 2 ' determine even or odd iteration number rotate additional +-90 (or +-45 for 4 iterations, which would otherwise only be +-90)

            if (n = 0) and (iterations <> 4) then
                ' if even AND iterations not equal to 4:
                JumpPol radius, theta + iterRotation
```

```
                MarkPol radius, -theta + iterRotation
            elseif (n = 0) and (iterations = 4) then
                ' if even AND iterations equal to 4:
                JumpPol radius, theta + iterRotation + pi/4
                MarkPol radius, -theta + iterRotation + pi/4
            else
                ' if odd:
                JumpPol radius, theta + iterRotation + pi/2
                MarkPol radius, -theta + iterRotation + pi/2
            end if
        next
        Call translateFocus(k,str)
    next
end sub


'*** translate focus depth ***
sub translateFocus(currentIter, str)
    if ((currentIter*ablationRate + currentFocus) > focusAdjust) then 'if current
     z value is greater than focusAdjust value, then adjust focus
        MoveRel ZA, -(currentIter*ablationRate + currentFocus)
        currentFocus = -currentIter*ablationRate ' save new current focus height
        echo str & currentFocus
        WaitUntilInPos ZA
        Wait 100
        echo "New ZA.pos: " & ZA.pos
    end if
end sub



'*** return to focus home ***
sub returnFocusHome(str)
    currentFocus = 0
    echo str & currentFocus
    Move ZA, focusHome
    WaitUntilInPos ZA
    Wait 100
end sub



'*** Print text  ***
sub printer(str)
    PreviewPenColor RGB(255, 255, 0) ' only relevant for laser GUI
    fnt.Height = labelSize
  print fnt, str
end sub



'*** VB initialization ***
sub loadInitialization
    'Preview settings (pensize and color)
    previewpensize 0.001
    PreviewPenColor RGB(0, 255, 0)

    ' font type and size
    Set fnt = CreateFont("din1451h")
    labelSize = 0.1 'mm

    ' calculate pi
    pi = 4 * Atn(1)
end sub
```