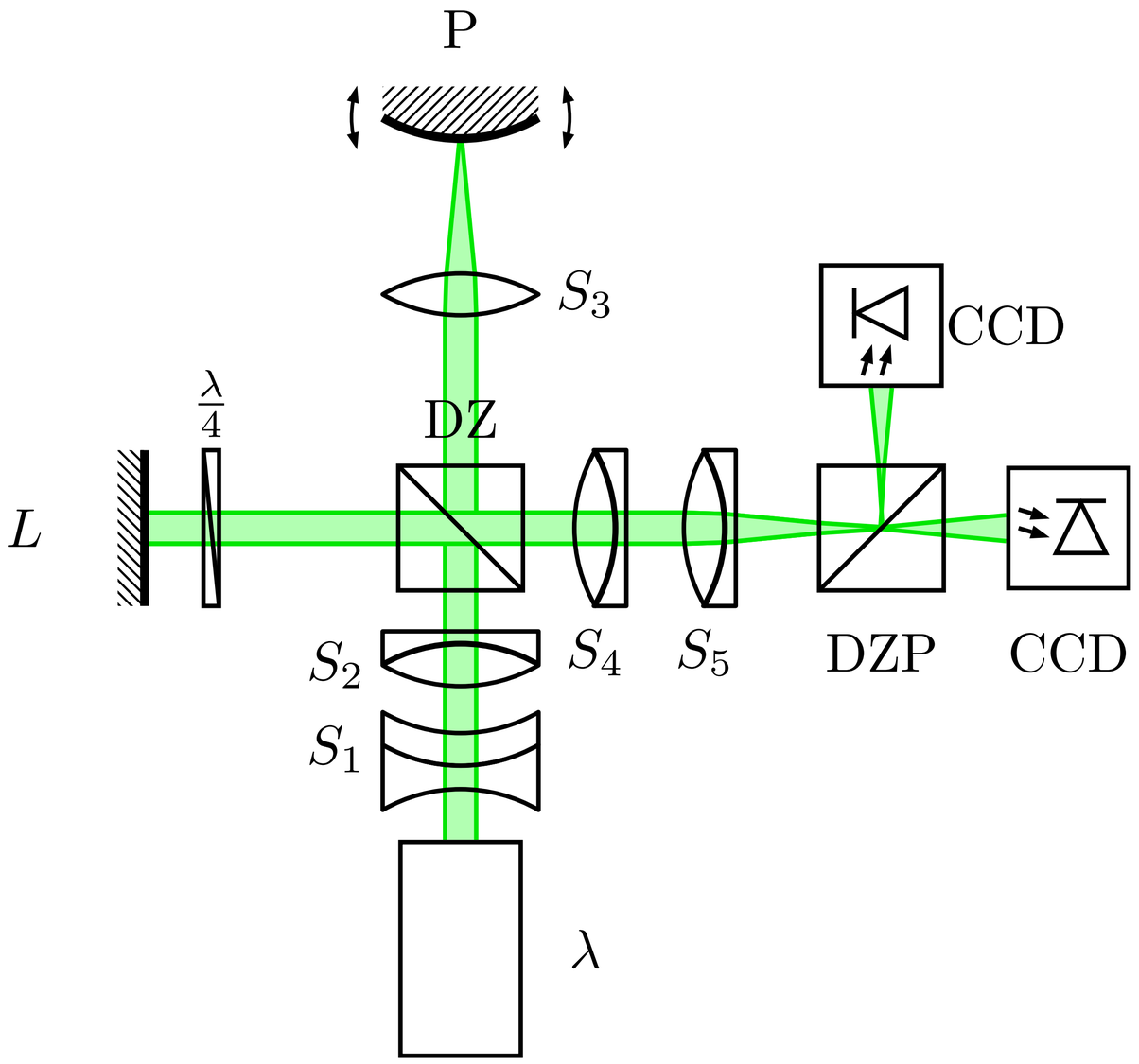
Programowanie systemów pomiarowych
w.3

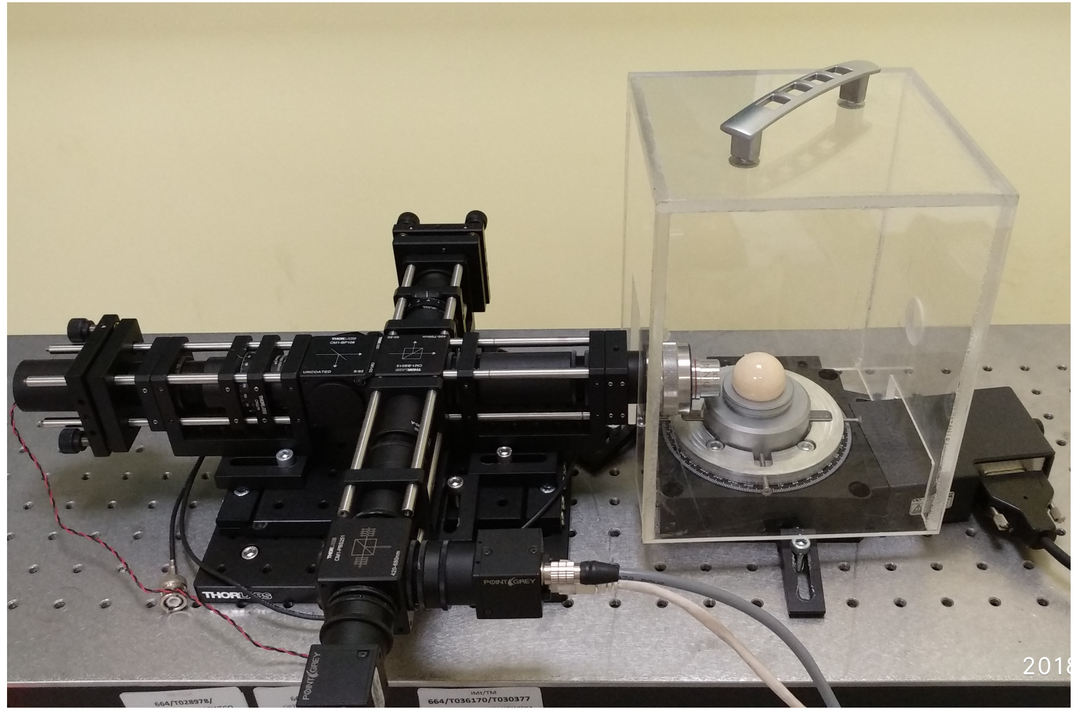
dr inż. Dawid Kucharski

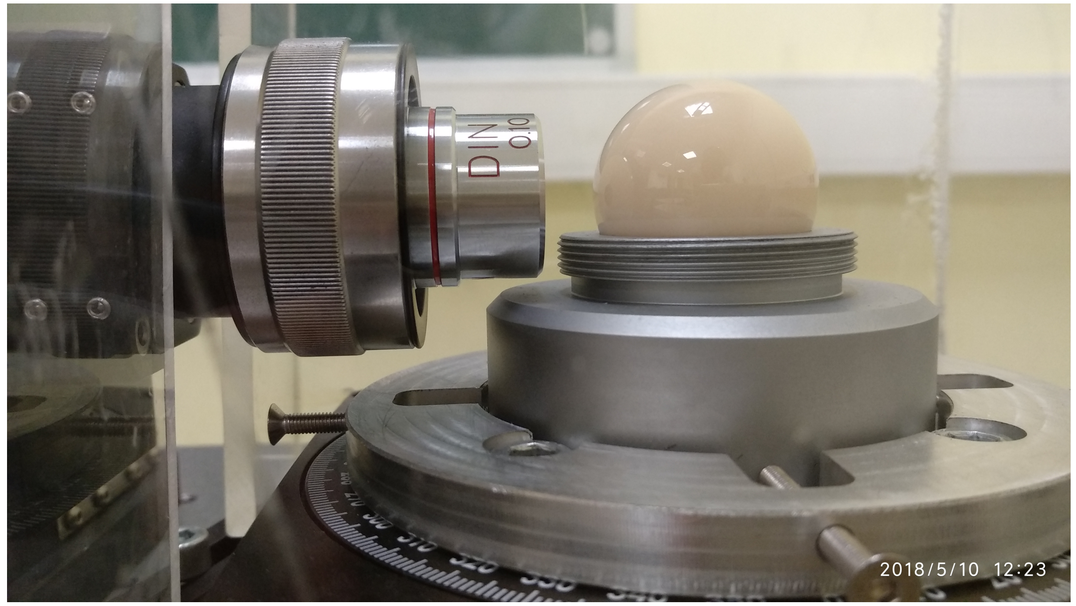
Zakład Metrologii i Systemów Pomiarowych
Instytut Technologii Mechanicznej
Wydział Budowy Maszyn i Zarządzania
Politechnika Poznańska

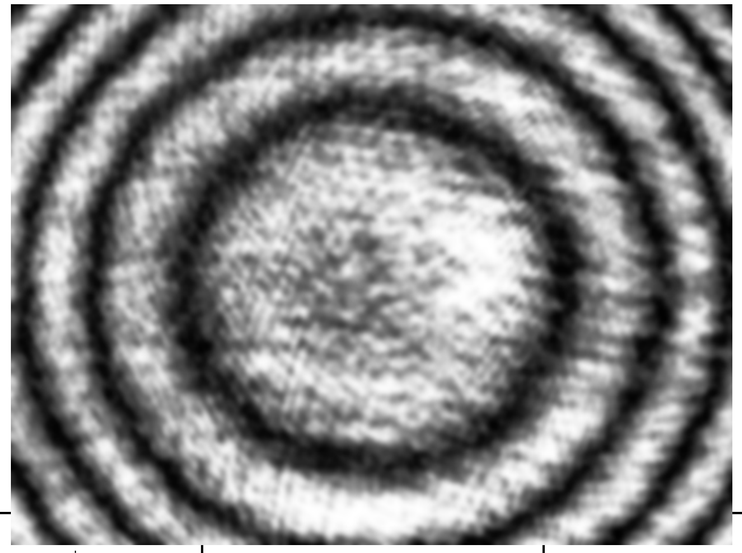
19.02.2019 13:12

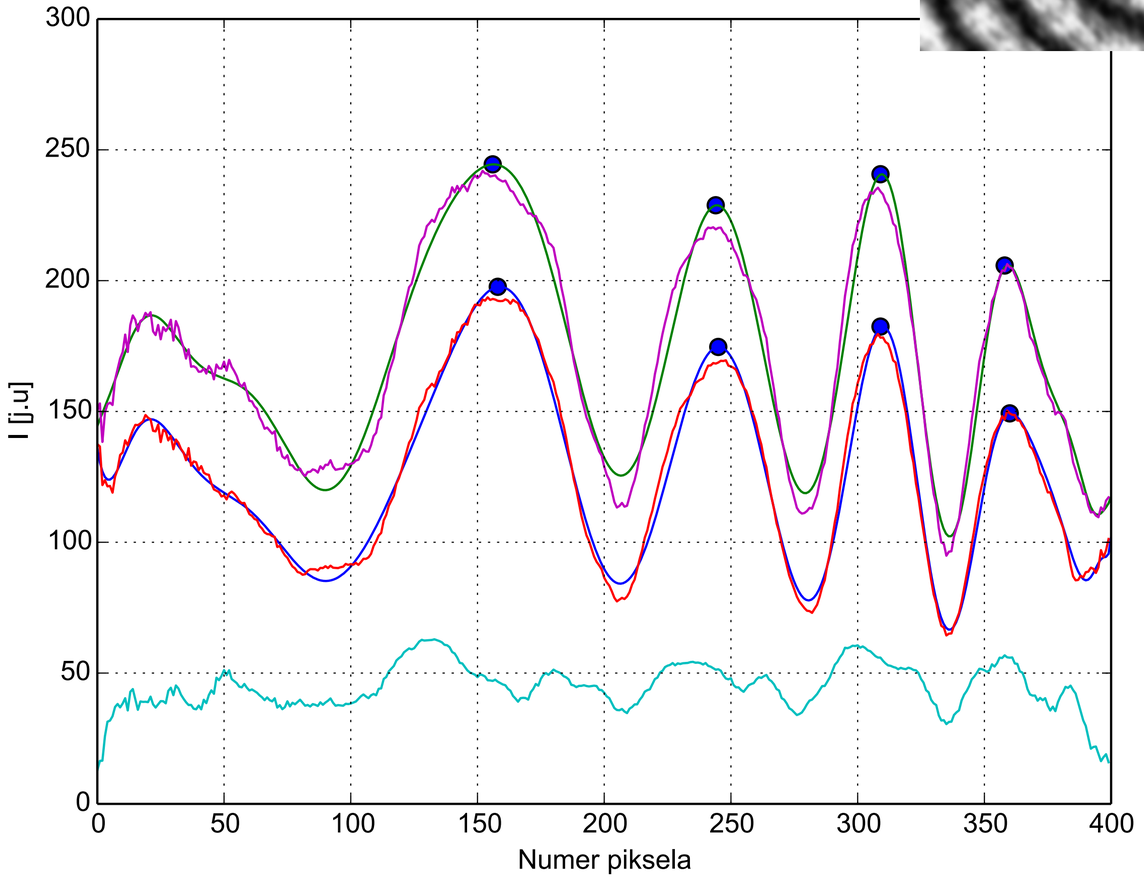
Zmodyfikowany interferometr T-G

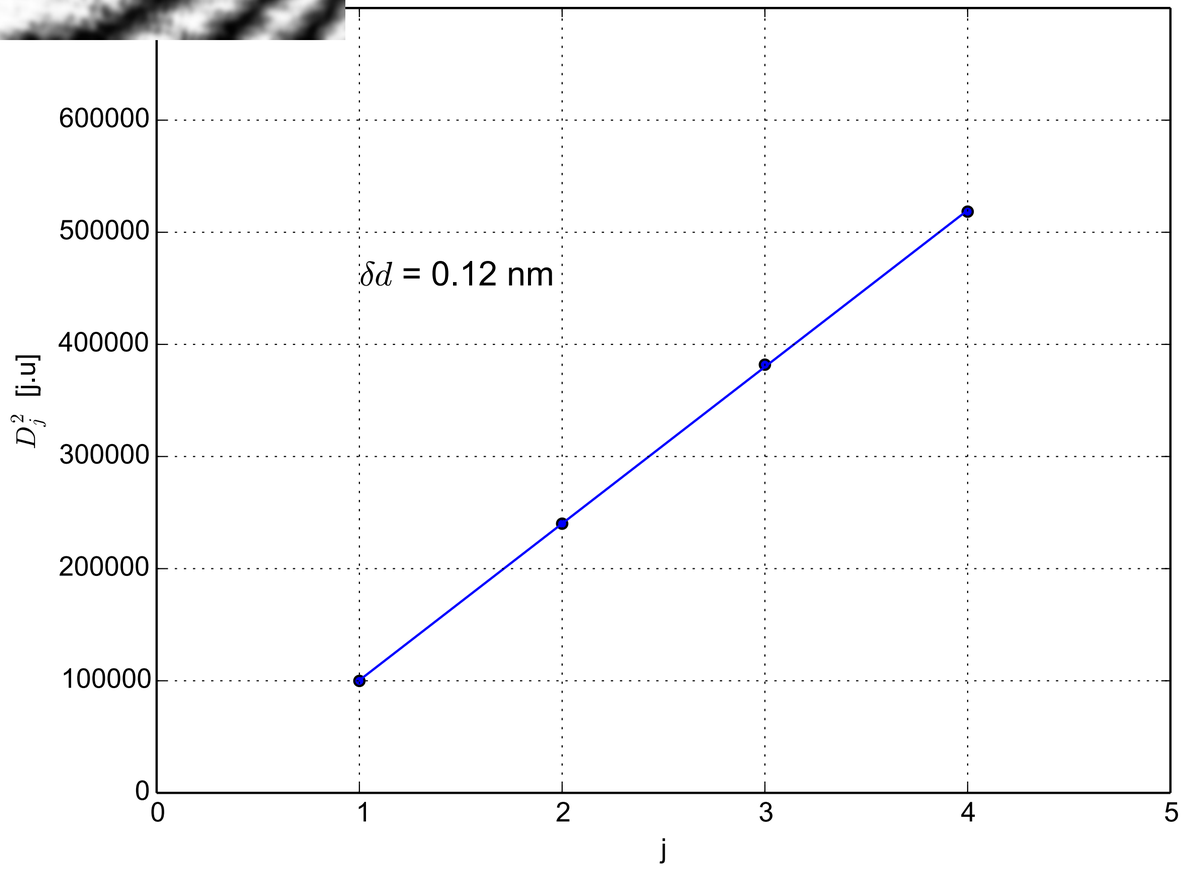




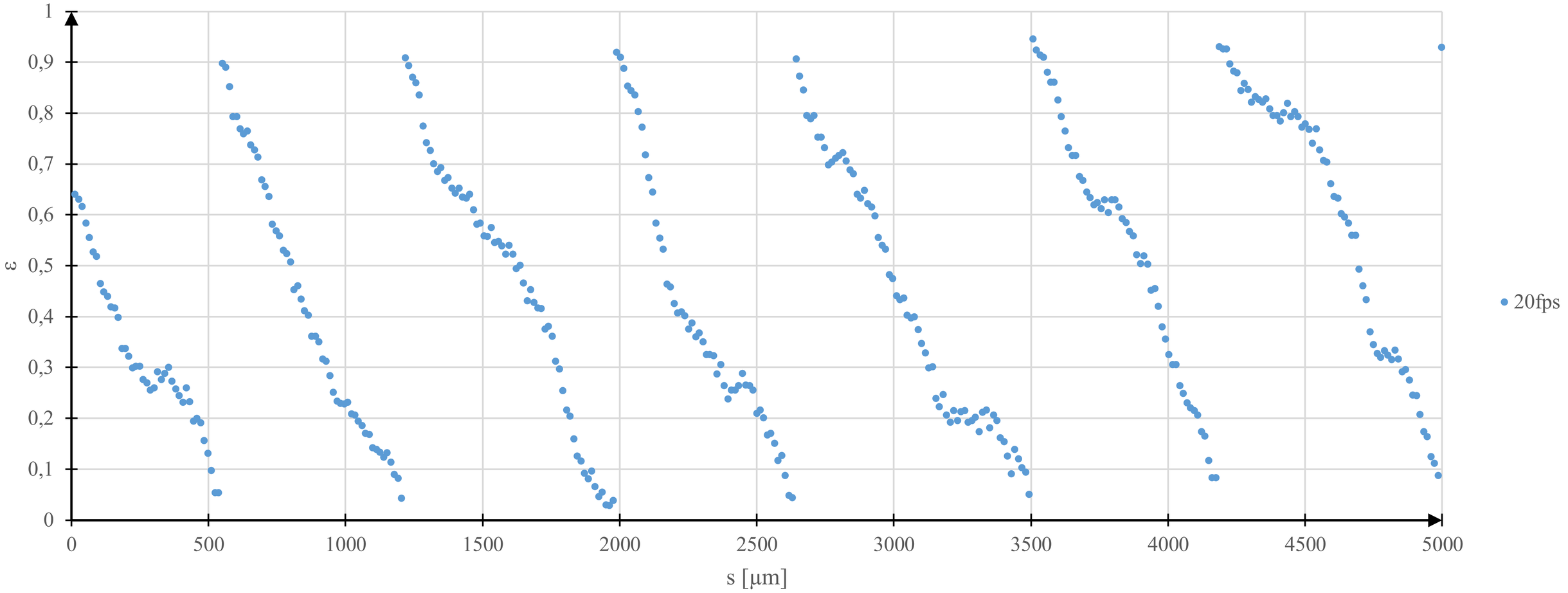


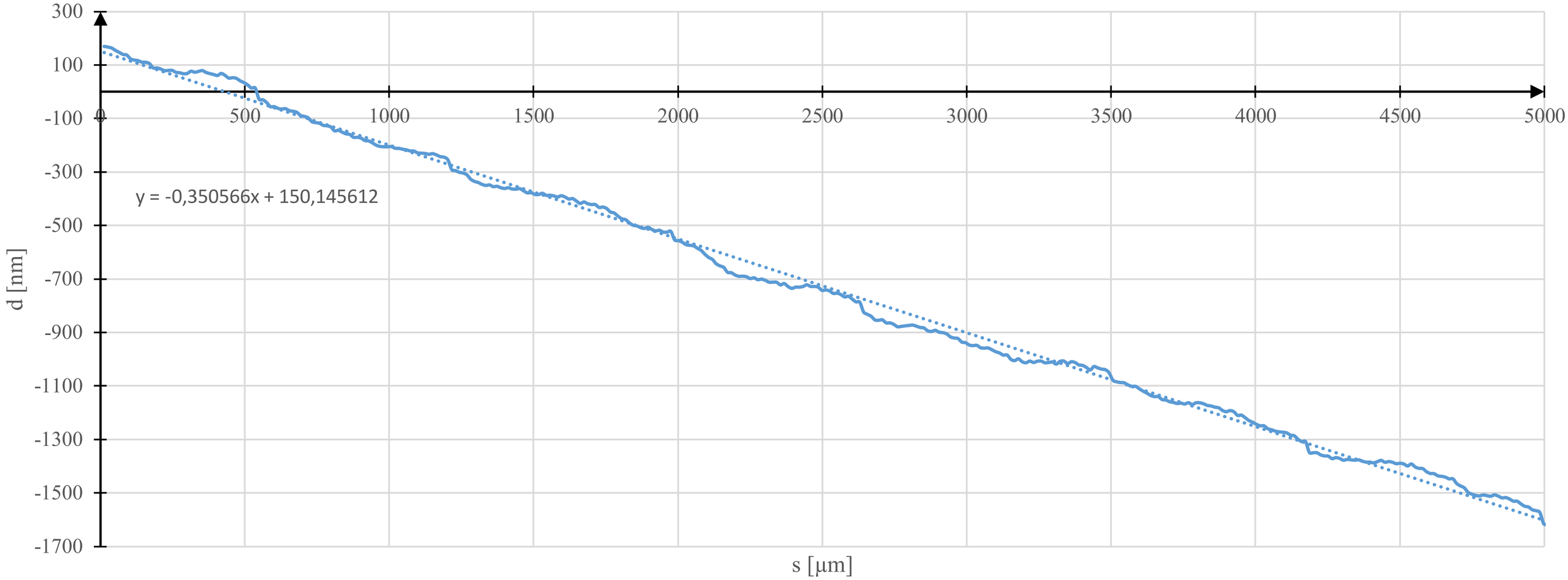


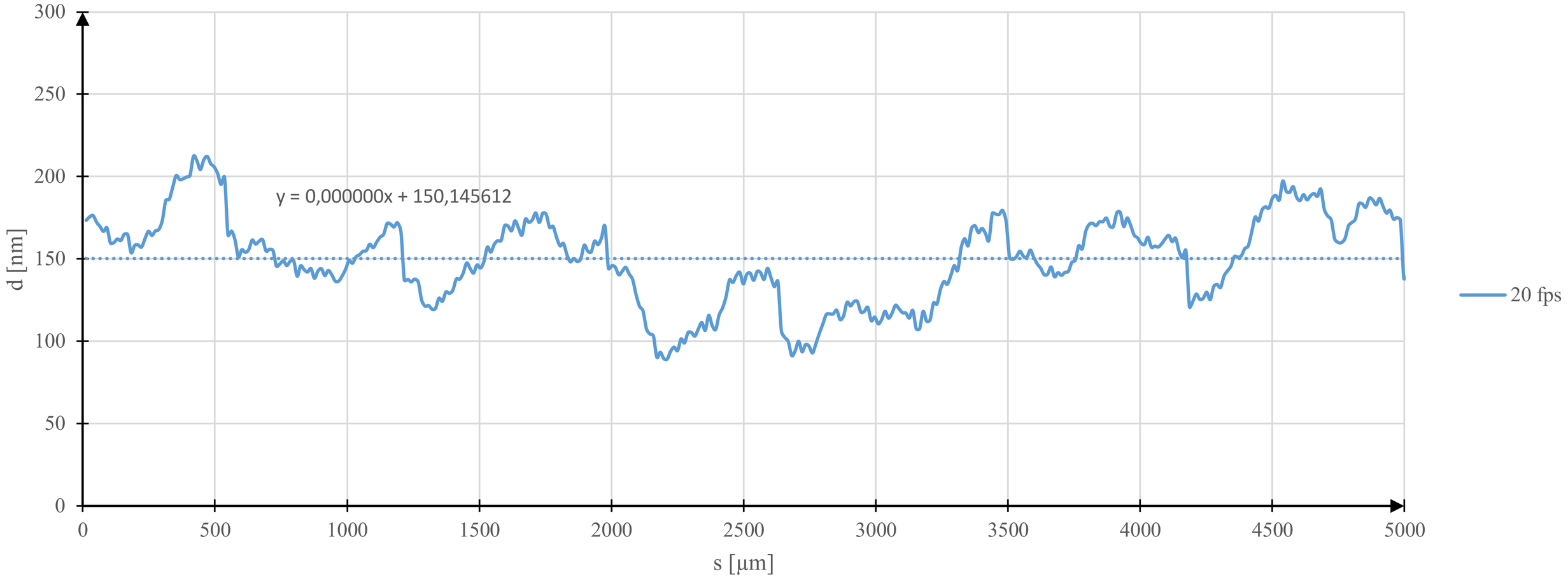

I [j.u]
Numer piksela
0
50
100
150
200
250
300
350
400

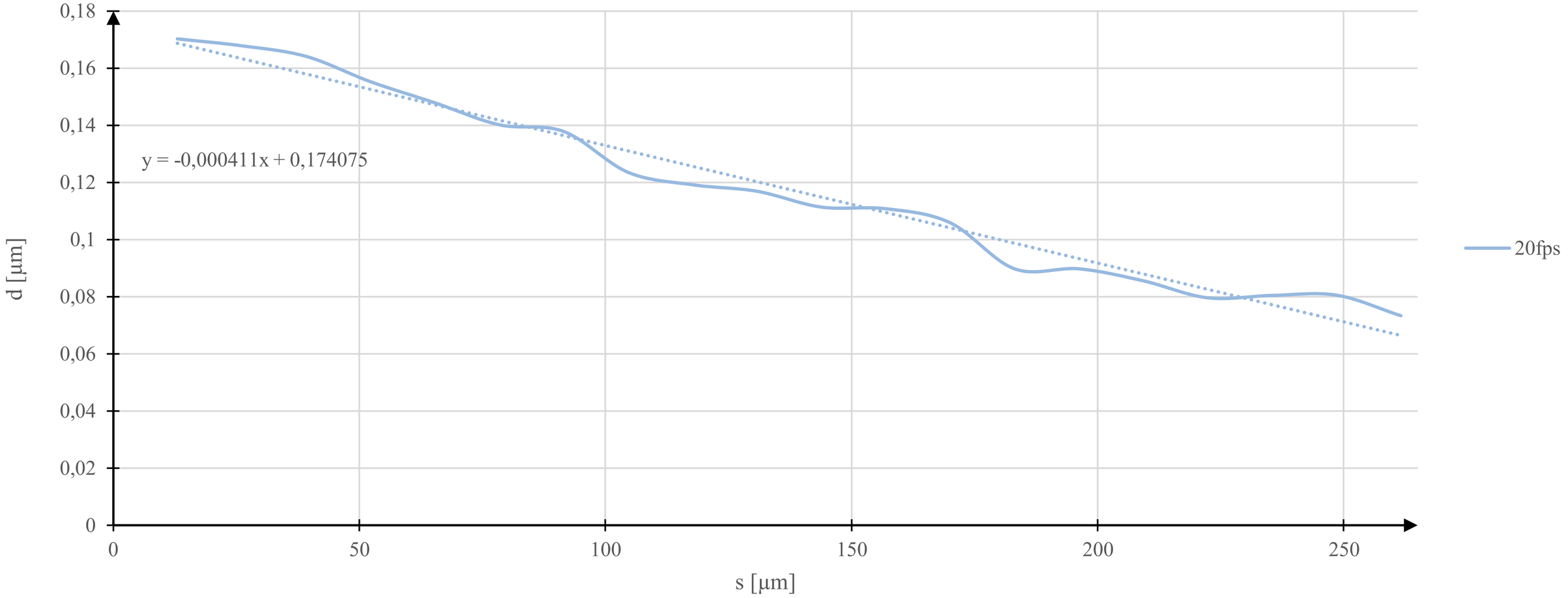

D_j^2 [j.u]
j
δd = 0.12 nm
0
100000
200000
300000
400000
500000
600000
1
2
3
4
5

- ceramiczny wzorzec okrągłości ϕ = 29, 9588 mm;
- odcinek 5000 µm (5% całego obwodu);
- 383 punkty pomiarowe;
- 20 fps;
- ω=1∘/s;
- α =400∘;
- cakowita liczba obrazów 8000.
- Odcinek elementarny 262 µm;
- 20 punktów pomiarowych.


1
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0
ε
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000
s [µm]
20fps

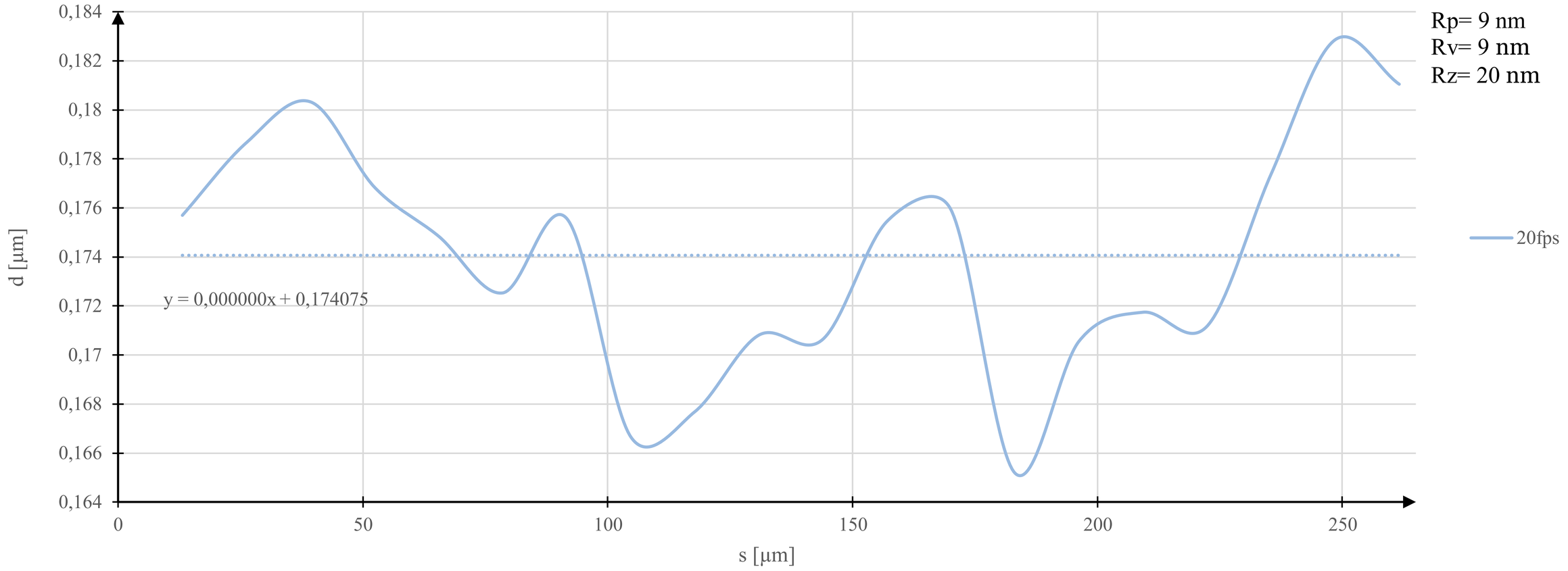

300
100
-100
-300
-500
-700
-900
-1100
-1300
-1500
-1700
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000
y = -0,350566x + 150,145612
d [nm]
s [µm]


300
250
200
150
100
50
0
d [nm]
y = 0,000000x + 150,145612
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000
s [µm]
20 fps


0,18
0,16
0,14
0,12
0,1
0,08
0,06
0,04
0,02
0
d [µm]
y = -0,000411x + 0,174075
0
50
100
150
200
250
s [µm]
20fps

Rp= 0,009 µm
Rv= 0,009 µm
Rz= 0,02 µm

Rp= 9 nm
Rv= 9 nm
Rz= 20 nm



Programowanie optycznego systemu pomiarowego

1. Programowanie pracy stołu obrotowego.
2. Programowanie pracy kamer CCD.
3. Program analizy danych pomiarowych.

Programowanie pracy stołu obrotowego w C+

```c
#include <stdio.h>
#include <fcntl.h>   /* File Control Definitions           */
#include <termios.h> /* POSIX Terminal Control Definitions */
#include <unistd.h>  /* UNIX Standard Definitions     */
#include <errno.h>   /* ERROR Number Definitions           */

void main(void)
{
   int fd;/*File Descriptor*/

printf("\n +----------------------------------
      +");
printf("\n |        Serial Port Write
      |");
printf("\n +----------------------------------
      +");

/*------------------------------ Opening the
Serial Port ------------------------------
*/

/* Change /dev/ttyUSB0 to the one
corresponding to your system */

     fd = open("/dev/ttyUSB0",O_RDWR |
          O_NOCTTY | O_NDELAY); /* ttyUSB0
          is the FT232 based USB2SERIAL
          Converter   */
                  /* O_RDWR Read/Write access to
                     serial port           */
                  /* O_NOCTTY - No terminal will control
                  the process   */
                  /* O_NDELAY -Non Blocking Mode,Does
                  not care about-  */
                  /* -the status of DCD line,Open()
                                  returns
                                  immedia
                                  tly */

     if(fd == -1)              /* Error Checking */
            printf("\n  Error! in Opening
                  ttyUSB0  ");
     else
            printf("\n  ttyUSB0 Opened
                  Successfully ");


/*---------- Setting the Attributes of the
serial port using termios structure ---------
*/

struct termios SerialPortSettings; /* Create
the structure                          */

tcgetattr(fd, &SerialPortSettings);   /* Get
        the current attributes of the
        Serial port */

cfsetispeed(&SerialPortSettings,B19200); /*
          Set Read  Speed as 9600
          */
cfsetospeed(&SerialPortSettings,B19200); /*
          Set Write Speed as 9600
          */

SerialPortSettings.c_cflag &= ~PARENB;   /*
```

```
    SerialPortSettings.c_cflag &= ~PARENB;   /*
                                  Disables the
                                  Parity Enable
                                  bit(PARENB),So
                                  No Parity   */
    SerialPortSettings.c_cflag &= ~CSTOPB;   /*
                                  CSTOPB = 2 Stop
                                  bits,here it is
                                  cleared so 1
                                  Stop bit */
    SerialPortSettings.c_cflag &= ~CSIZE;  /*
                                  Clears the mask
                                  for setting the
                                  data size
                                  */
    SerialPortSettings.c_cflag |=  CS8;      /*
                                   Set the data
                                   bits = 8
                                   */

    SerialPortSettings.c_cflag |= CRTSCTS;
                                  /* Enable
                                  Hardware flow
                                  Control
                                  */
    SerialPortSettings.c_cflag |= CREAD | CLOCAL;
                                  /* Enable
                                  receiver,Ignore
                                  Modem Control
                                  lines       */


    SerialPortSettings.c_iflag &= ~(IXON | IXOFF
                                  | IXANY);
                                  /* Disable
                                  XON/XOFF flow
                                  control both
                                  i/p and o/p */
    SerialPortSettings.c_iflag &= ~(ICANON | ECHO
                                  | ECHOE | ISIG);
                                  /* Non
                                  Cannonical mode
                                  */

    SerialPortSettings.c_oflag &= ~OPOST;/*No
                                  Output
                                  Processing*/

    /* Setting Time outs */
    SerialPortSettings.c_cc[VMIN] = 10; /* Read
                                   at least 10
                                   characters
                                   */
    SerialPortSettings.c_cc[VTIME] = 0; /* Wait
                                    indefinetly
                                    */

    if((tcsetattr(fd,TCSANOW,&SerialPortSettings))
      != 0) /* Set the attributes to the termios
      structure*/
         printf("\n  ERROR ! in Setting
               attributes");
    else
                    printf("\n  BaudRate =
                          19200 \n  StopBits =
                          1 \n  Parity    =
                          none");

          /*-------------------------------
          Write data to serial port -------------
          ----------------*/

    /* "write_buffer" - Buffer containing
    characters to write into port       */
```

```
143     characters to write into port      */
144     /* "bytes_written" - Value for storing the
145     number of bytes written to the port */
146     /* use write() to send data to port
147                */
148     /* "fd" - file descriptor pointing to the
149     opened serial port */
150          /* "write_buffer" - address of the
151          buffer containing data     */
152           /* "sizeof(write_buffer)" - No of
153                       bytes to write */
154
155     char write_buffer1[] = "1AC5;WT60\r";
156                           /*Set acceleration in
157                           degrees/s^2 */
158     int bytes_written  = 0;
159
160     bytes_written = write(fd,write_buffer1,sizeof(
161                     write_buffer1));
162     printf("\n  1AC5 written to ttyUSB0");
163     printf("\n +----------------------------------
164          +\n\n");
165
166     char write_buffer2[] = "1AG5;WT60\r";
167                           /*Set deceleration in
168                           degrees/s^2 */
169     bytes_written  = 0;
170
171     bytes_written = write(fd,write_buffer2,sizeof(
172                     write_buffer2));
173     printf("\n  1AG5 written to ttyUSB0");
174     printf("\n +----------------------------------
175          +\n\n");
176
177     char write_buffer3[] = "1VA0.4;WT60\r";
178                           /*Set velocity in
179                           degrees/s */
```

```
180     bytes_written  = 0;
181
182     bytes_written = write(fd,write_buffer3,sizeof(
183                     write_buffer3));
184     printf("\n  1VA1 written to ttyUSB0");
185     printf("\n +----------------------------------
186          +\n\n");
187
188     char write_buffer4[] = "1MO;WT60\r";          /*
189                           Motor on  */
190     bytes_written  = 0;
191
192     bytes_written = write(fd,write_buffer4,sizeof(
193                     write_buffer4));
194     printf("\n  1MO written to ttyUSB0");
195     printf("\n +----------------------------------
196          +\n\n");
197
198     char write_buffer5[] = "1OR1;WT18000\r";        /*
199                           Search for home and
200                           wait 18 seconds */
201     bytes_written  = 0;
202
203     bytes_written = write(fd,write_buffer5,sizeof(
204                     write_buffer5));
205     printf("\n  1OR1 written to ttyUSB0");
206     printf("\n +----------------------------------
207          +\n\n");
208
209     char write_buffer6[] = "1PR800;WT60\r";         /*
210                           Move to absolute
211                           position 800 degrees
212                           */
213     bytes_written  = 0;
214
215     bytes_written = write(fd,write_buffer6,sizeof(
216                     write_buffer6));
```

```
217     printf("\n  1WP400 written to ttyUSB0");
218     printf("\n +----------------------------------
219          +\n\n");
220
221     char write_buffer7[] = "1WP800;WT60\r";          /*
222                           Wait for position 800
223                           */
224     bytes_written  = 0;
225
226     bytes_written = write(fd,write_buffer7,sizeof(
227                     write_buffer7));
228     printf("\n  1PR400 written to ttyUSB0");
229     printf("\n +----------------------------------
230          +\n\n");
231
232
233     char write_buffer8[] = "1AC2;WT60\r";
234                           /*Set acceleration in
235                           degrees/s^2 */
236     bytes_written  = 0;
237
238     bytes_written = write(fd,write_buffer8,sizeof(
239                     write_buffer8));
240     printf("\n  1AC2 written to ttyUSB0");
241     printf("\n +----------------------------------
242          +\n\n");
243
244     char write_buffer9[] = "1AG2;WT60\r";
245                           /*Set deceleration in
246                           degrees/s^2 */
247     bytes_written  = 0;
248
249     bytes_written = write(fd,write_buffer9,sizeof(
250                     write_buffer9));
251     printf("\n  1AG2 written to ttyUSB0");
252     printf("\n +----------------------------------
253          +\n\n");
254
255     char write_buffer10[] = "1VA40;WT6000\r";
256                           /*Set velocity in
257                           degrees/s */
258     bytes_written  = 0;
259
260     bytes_written = write(fd,write_buffer10,
261                     sizeof(write_buffer10));
262     printf("\n  1VA40 written to ttyUSB0");
263     printf("\n +----------------------------------
264          +\n\n");
265
266     char write_buffer11[] = "1PR-800;WT60\r";
267                           /* Move to absolute
268                           position 0 degrees
269                           */
270     bytes_written  = 0;
271
272     bytes_written = write(fd,write_buffer11,
273                     sizeof(write_buffer11));
274     printf("\n  1WP-400 written to ttyUSB0");
275     printf("\n +----------------------------------
276          +\n\n");
277
278     char write_buffer12[] = "1WP0;WT60\r";           /*
279                           Wait for position 0
280                           */
281     bytes_written  = 0;
282
283     bytes_written = write(fd,write_buffer12,
284                     sizeof(write_buffer12));
285     printf("\n  1PR0 written to ttyUSB0");
286     printf("\n +----------------------------------
287          +\n\n");
288
289     char write_buffer13[] = "WT60\r";           /*  Stop
290                           motor */
```

```
291     bytes_written  = 0;
292
293     bytes_written = write(fd,write_buffer13,
294                   sizeof(write_buffer13));
295     printf("\n  1WP0 written to ttyUSB0");
296     printf("\n +----------------------------------
297          +\n\n");
298
299     char write_buffer14[] = "1MF\r";         /*  Motor
300                            off */
301     bytes_written  = 0;
302
303     bytes_written = write(fd,write_buffer14,
304                   sizeof(write_buffer14));
305     printf("\n 1ST written to ttyUSB0");/*
306     printf("\n +----------------------------------
307          +\n\n");
308
309
310
311
312
313     close(fd);/* Close the Serial port */
314
315        }
316
```

Programowanie pracy kamer CCD

```
#include "stdafx.h"

#include "FlyCapture2.h"
#include "sys/time.h"
#include "sstream"
using namespace FlyCapture2;

void PrintBuildInfo()
{
    FC2Version fc2Version;
    Utilities::GetLibraryVersion( &fc2Version );
    char version[128];
    sprintf(
        version,
        "FlyCapture2 library version: %d.%d.%d.
        %d\n",
        fc2Version.major, fc2Version.minor,
        fc2Version.type, fc2Version.build );

    printf( version );

    char timeStamp[512];
    sprintf( timeStamp, "Application build date:
             %s %s\n\n", __DATE__, __TIME__ );

    printf( timeStamp );
}

void PrintCameraInfo( CameraInfo* pCamInfo )
{

}

void PrintError( Error error )
```

```
void PrintError( Error error )
{
    error.PrintErrorTrace();
}

int RunSingleCamera( PGRGuid guid )
{
    const int k_numImages = 10000;

    Error error;
    Camera cam;

    // Connect to a camera
    error = cam.Connect(&guid);
    if (error != PGRERROR_OK)
    {
        PrintError( error );
        return -1;
    }

    // Get the camera information
    CameraInfo camInfo;
    error = cam.GetCameraInfo(&camInfo);
    if (error != PGRERROR_OK)
    {
        PrintError( error );
        return -1;
    }

    PrintCameraInfo(&camInfo);


    // Start capturing images
    error = cam.StartCapture();


    Image rawImage;
```

```
71
72
73      for ( int imageCnt=0; imageCnt <
74            k_numImages; imageCnt++ )
75      {
76          // Retrieve an image
77          error = cam.RetrieveBuffer( &rawImage );
78     struct timeval now;
79          //printf( "Grabbed image %d\n",
80                    imageCnt);
81  gettimeofday (&now, NULL); //get current time
82  long long milliseconds = now.tv_sec*1000LL +
83                           now.tv_usec/1000; //
84                           calculate
85                           milliseconds;
86  printf ("2CH image %d milliseconds:%lld\n",
87           imageCnt, milliseconds);
88          // Create a converted image
89          Image convertedImage;
90
91          // Convert the raw image
92          error = rawImage.Convert(
93                  PIXEL_FORMAT_MONO8,
94                  &convertedImage );
95
96          // Create a unique filename
97          char filename[512];
98          sprintf( filename,
99                   "/home/dawid/11042018/rotation_
100                  meas/04deg_per_s_10fps_10000fra
101                  mes#2/2channel/%04d.png",
102                  imageCnt);
103
104         // Save the image. If a file format is
105         not passed in, then the file
106         // extension is parsed to attempt to
107         determine the file format.
108             error = convertedImage.Save( filename );
109
110
111     }
112
113
114     // Stop capturing images
115     error = cam.StopCapture();
116
117
118     // Disconnect the camera
119     error = cam.Disconnect();
120
121
122     return 0;
123 }
124
125 int main(int /*argc*/, char** /*argv*/)
126 {
127     PrintBuildInfo();
128
129     Error error;
130
131     // Since this application saves images in
132     the current folder
133     // we must ensure that we have permission
134     to write to this folder.
135     // If we do not have permission, fail right
136     away.
137  FILE* tempFile = fopen("test.txt", "w+");
138  if (tempFile == NULL)
139  {
140     printf("Failed to create file in current
141           folder.  Please check permissions.\n");
142     return -1;
143  }
144  fclose(tempFile);
```

```
145  remove("test.txt");
146
147     BusManager busMgr;
148     unsigned int numCameras;
149     error = busMgr.GetNumOfCameras(&numCameras);
150
151
152
153
154
155         PGRGuid guid;
156         error = busMgr.GetCameraFromIndex(1,
157                 &guid);
158
159         RunSingleCamera( guid );
160
161
162
163
164
165
166 }
```

Uruchamianie pomiaru - #Bash

```
1 #!/bin/bash
2 ./standard && sleep 25 &&
3 shopt -s extblob
4 ./channel1 & PIDIOS=$!
5 ./channel2 & PIDMIX=$!
6 wait $PIDIOS
7 wait $PIDMIX
```

```
1 #!/bin/bash
2 ./standard && sleep 25 &&
3 shopt -s extblob
4 ./channel1 & PIDIOS=$!
5
6 wait $PIDIOS
7 wait $PIDMIX
```

```
1 #!/bin/bash
2 until date | fgrep -q "20:00:"; do
3 sleep 10
4 done
5 ./standard && sleep 25 &&
6 shopt -s extblob
7 ./channel1 & PIDIOS=$!
8 wait $PIDIOS
9 wait $PIDMIX
```

Analiza obrazów interferencyjnych - Python

```python
import numpy as np
import warnings
warnings.filterwarnings('ignore')

def azimuthalAverage(image, center=None,
                     stddev=False,
                     returnradii=False,
                     return_nr=False,
        binsize=0.5, weights=None, steps=False,
                interpnan=False, left=None,
                right=None,
        mask=None ):
    """
    Calculate the azimuthally averaged radial
    profile.

    image - The 2D image
    center - The [x,y] pixel coordinates used
    as the center. The default is
             None, which then uses the center
             of the image (including
             fractional pixels).
    stddev - if specified, return the azimuthal
    standard deviation instead of the average
    returnradii - if specified, return (radii_ar
                                        ray,
                                        radial_p
                                        rofile)
    return_nr   - if specified, return number
    of pixels per radius *and* radius
    binsize - size of the averaging bin.  Can
    lead to strange results if
        non-binsize factors are used to specify
        the center and the binsize is
        too large
    weights - can do a weighted average instead
    of a simple average if this keyword
    parameter
        is set.  weights.shape must = image.
                                     shape.
                                     weighted
                                     stddev is
                                     undefined,
                                     so don't
        set weights and stddev.
    steps - if specified, will return a double-
    length bin array and radial
        profile so you can plot a step-form
        radial profile (which more accurately
        represents what's going on)
    interpnan - Interpolate over NAN values, i.
    e. bins where there is no data?
        left,right - passed to interpnan; they
        set the extrapolated values
    mask - can supply a mask (boolean array
                              same size as
                              image with True
                              for OK and False
                              for not)
        to average over only select data.

    If a bin contains NO DATA, it will have a
    NAN value because of the
    divide-by-sum-of-weights component.  I
    think this is a useful way to denote
    lack of data, but users let me know if an
    alternative is prefered...

    """
    # Calculate the indices from the image
    y, x = np.indices(image.shape)
```

```
72
73      if center is None:
74          center = np.array([(x.max()-x.min())/2.
75                  0, (y.max()-y.min())/2.0])
76
77      r = np.hypot(x - center[0], y - center[1])
78
79      if weights is None:
80          weights = np.ones(image.shape)
81      elif stddev:
82          raise ValueError("Weighted standard
83                           deviation is not
84                           defined.")
85
86      if mask is None:
87          mask = np.ones(image.shape,dtype='bool')
88      # obsolete elif len(mask.shape) > 1:
89      # obsolete     mask = mask.ravel()
90
91      # the 'bins' as initially defined are
92      lower/upper bounds for each bin
93      # so that values will be in [lower,upper)
94      nbins = int(np.round(r.max() / binsize)+1)
95      maxbin = nbins * binsize
96      bins = np.linspace(0,maxbin,nbins+1)
97      # but we're probably more interested in the
98      bin centers than their left or right sides..
99      .
100     bin_centers = (bins[1:]+bins[:-1])/2.0
101
102     # how many per bin (i.e., histogram)?
103     # there are never any in bin 0, because the
104     lowest index returned by digitize is 1
105     #nr = np.bincount(whichbin)[1:]
106     nr = np.histogram(r,bins)[0]
107
108     # recall that bins are from 1 to nbins (whic
109                                                 h
110                                                 is
111                                                 expr
112                                                 esse
113                                                 d
114                                                 in
115                                                 arra
116                                                 y
117                                                 term
118                                                 s
119                                                 by
120                                                 aran
121                                                 ge(
122                                                 nbin
123                                                 s)+
124                                                 1
125                                                 or
126                                                 xran
127                                                 ge(
128                                                 1,
129                                                 nbin
130                                                 s+1)
131                                                 )
132     # radial_prof.shape = bin_centers.shape
133     if stddev:
134         # Find out which radial bin each point
135         in the map belongs to
136         whichbin = np.digitize(r.flat,bins)
137         # This method is still very slow; is
138         there a trick to do this with
139         histograms?
140         radial_prof = np.array([image.flat[mask.
141                         flat*(whichbin==b)].std()
142                         for b in xrange(1,nbins+1)
143                         ])
144     else:
145         radial_prof = np.histogram(r, bins,
```

```
                    weights=(
                    image*weights*mask))[0] /
                    np.histogram(r, bins,
                    weights=(mask*weights))[0]

    if interpnan:
        radial_prof = np.interp(bin_centers,
                    bin_centers[
                    radial_prof==radial_prof],
                    radial_prof[
                    radial_prof==radial_prof],
                    left=left,right=right)

    if steps:
        xarr = np.array(zip(bins[:-1],bins[1:]))
               .ravel()
        yarr = np.array(zip(radial_prof,
               radial_prof)).ravel()
        return xarr,yarr
    elif returnradii:
        return bin_centers,radial_prof
    elif return_nr:
        return nr,bin_centers,radial_prof
    else:
        return radial_prof

def azimuthalAverageBins(image,azbins,
                         symmetric=None,
                         center=None, **kwargs):
    """ Compute the azimuthal average over a
    limited range of angles
    kwargs are passed to azimuthalAverage """
    y, x = np.indices(image.shape)
    if center is None:
        center = np.array([(x.max()-x.min())/2.
                 0, (y.max()-y.min())/2.0])
    r = np.hypot(x - center[0], y - center[1])
    theta = np.arctan2(x - center[0], y -
            center[1])
    theta[theta < 0] += 2*np.pi
    theta_deg = theta*180.0/np.pi

    if isinstance(azbins,np.ndarray):
        pass
    elif isinstance(azbins,int):
        if symmetric == 2:
            azbins = np.linspace(0,90,azbins)
            theta_deg = theta_deg % 90
        elif symmetric == 1:
            azbins = np.linspace(0,180,azbins)
            theta_deg = theta_deg % 180
        elif azbins == 1:
            return azbins,azimuthalAverage(image
                                           ,cent
                                           er=ce
                                           nter,
                                           retur
                                           nradi
                                           i=Tru
                                           e,
                                           **kwa
                                           rgs)
        else:
            azbins = np.linspace(0,359.
                                 9999999999999,azbins)
    else:
        raise ValueError("azbins must be an
                         ndarray or an integer")

    azavlist = []
    for blow,bhigh in zip(azbins[:-1],azbins[1:]
                          ):
        mask = (theta_deg > (blow % 360)) * (
               theta_deg < (bhigh % 360))
```

```
220         rr,zz = azimuthalAverage(image,
221                 center=center,mask=mask,
222                 returnradii=True,**kwargs)
223         azavlist.append(zz)
224 
225     return azbins,rr,azavlist
226 
227 def radialAverage(image, center=None,
228                   stddev=False, returnAz=False,
229                   return_naz=False,
230         binsize=1.0, weights=None, steps=False,
231                 interpnan=False, left=None,
232                 right=None,
233         mask=None, symmetric=None ):
234     """
235     Calculate the radially averaged azimuthal
236     profile.
237     (this code has not been optimized; it could
238      be speed boosted by ~20x)
239 
240     image - The 2D image
241     center - The [x,y] pixel coordinates used
242     as the center. The default is
243              None, which then uses the center
244              of the image (including
245              fractional pixels).
246     stddev - if specified, return the radial
247     standard deviation instead of the average
248     returnAz - if specified, return (azimuthArra
249                                      y,
250                                      azimuthal_p
251                                      rofile)
252     return_naz   - if specified, return number
253     of pixels per azimuth *and* azimuth
254     binsize - size of the averaging bin.  Can
255     lead to strange results if
256         non-binsize factors are used to specify
257         the center and the binsize is
258         too large
259     weights - can do a weighted average instead
260     of a simple average if this keyword
261     parameter
262         is set.  weights.shape must = image.
263                                      shape.
264                                      weighted
265                                      stddev is
266                                      undefined,
267                                      so don't
268         set weights and stddev.
269     steps - if specified, will return a double-
270     length bin array and azimuthal
271         profile so you can plot a step-form
272         azimuthal profile (which more
273         accurately
274         represents what's going on)
275     interpnan - Interpolate over NAN values, i.
276     e. bins where there is no data?
277         left,right - passed to interpnan; they
278         set the extrapolated values
279     mask - can supply a mask (boolean array
280                               same size as
281                               image with True
282                               for OK and False
283                               for not)
284         to average over only select data.
285 
286     If a bin contains NO DATA, it will have a
287     NAN value because of the
288     divide-by-sum-of-weights component.  I
289     think this is a useful way to denote
290     lack of data, but users let me know if an
291     alternative is prefered...
292 
293     """
```

```
    # Calculate the indices from the image
    y, x = np.indices(image.shape)

    if center is None:
        center = np.array([(x.max()-x.min())/2.
                  0, (y.max()-y.min())/2.0])

    r = np.hypot(x - center[0], y - center[1])
    theta = np.arctan2(x - center[0], y -
            center[1])
    theta[theta < 0] += 2*np.pi
    theta_deg = theta*180.0/np.pi
    maxangle = 360

    if weights is None:
        weights = np.ones(image.shape)
    elif stddev:
        raise ValueError("Weighted standard
                         deviation is not
                         defined.")

    if mask is None:
        # mask is only used in a flat context
        mask = np.ones(image.shape,dtype='bool')
               .ravel()
    elif len(mask.shape) > 1:
        mask = mask.ravel()

    # allow for symmetries
    if symmetric == 2:
        theta_deg = theta_deg % 90
        maxangle = 90
    elif symmetric == 1:
        theta_deg = theta_deg % 180
        maxangle = 180

    # the 'bins' as initially defined are
    lower/upper bounds for each bin
    # so that values will be in [lower,upper)
    nbins = int(np.round(maxangle / binsize))
    maxbin = nbins * binsize
    bins = np.linspace(0,maxbin,nbins+1)
    # but we're probably more interested in the
    bin centers than their left or right sides..
    .
    bin_centers = (bins[1:]+bins[:-1])/2.0

    # Find out which azimuthal bin each point
    in the map belongs to
    whichbin = np.digitize(theta_deg.flat,bins)

    # how many per bin (i.e., histogram)?
    # there are never any in bin 0, because the
    lowest index returned by digitize is 1
    nr = np.bincount(whichbin)[1:]

    # recall that bins are from 1 to nbins (whic
                                            h
                                            is
                                            expr
                                            esse
                                            d
                                            in
                                            arra
                                            y
                                            term
                                            s
                                            by
                                            aran
                                            ge(
                                            nbin
                                            s)+
                                            1
                                            or
```

```
                                        xran
                                        ge(
                                        1,
                                        nbin
                                        s+1)
                                        )
    # azimuthal_prof.shape = bin_centers.shape
    if stddev:
        azimuthal_prof = np.array([image.flat[
                         mask*(whichbin==b)].
                         std() for b in xrange(
                         1,nbins+1)])
    else:
        azimuthal_prof = np.array([(
                         image*weights).flat[
                         mask*(whichbin==b)].
                         sum() / weights.flat[
                         mask*(whichbin==b)].
                         sum() for b in xrange(
                         1,nbins+1)])

    #import pdb; pdb.set_trace()

    if interpnan:
        azimuthal_prof = np.interp(bin_centers,
            bin_centers[azimuthal_prof==azimutha
                                    l_prof],
            azimuthal_prof[azimuthal_prof==azimu
                                        thal_p
                                        rof],
            left=left,right=right)

    if steps:
        xarr = np.array(zip(bins[:-1],bins[1:]))
               .ravel()
        yarr = np.array(zip(azimuthal_prof,
               azimuthal_prof)).ravel()
    elif returnAz:
        return bin_centers,azimuthal_prof
    elif return_naz:
        return nr,bin_centers,azimuthal_prof
    else:
        return azimuthal_prof

def radialAverageBins(image,radbins,
                      corners=True, center=None,
                      **kwargs):
    """ Compute the radial average over a
    limited range of radii """
    y, x = np.indices(image.shape)
    if center is None:
        center = np.array([(x.max()-x.min())/2.
                  0, (y.max()-y.min())/2.0])
    r = np.hypot(x - center[0], y - center[1])

    if isinstance(radbins,np.ndarray):
        pass
    elif isinstance(radbins,int):
        if radbins == 1:
            return radbins,radialAverage(image,
                                         center=
                                         center,
                                         returnA
                                         z=True,
                                         **kwarg
                                         s)
        elif corners:
            radbins = np.linspace(0,r.max(),
                      radbins)
        else:
            radbins = np.linspace(0,np.max(np.
                      abs(np.array([x-center[0],
                      y-center[1]]))),radbins)
    else:
```

```
        raise ValueError("radbins must be an
                          ndarray or an integer")

    radavlist = []
    for blow,bhigh in zip(radbins[:-1],radbins[
                          1:]):
        mask = (r<bhigh)*(r>blow)
        az,zz = radialAverage(image,
                center=center,mask=mask,
                returnAz=True,**kwargs)
        radavlist.append(zz)

    return radbins,az,radavlist
import glob
path = '/home/dawid/11042018/rotation_meas/06deg
       _per_s_15fps_10000frames#2/2channel/*.
       png'
files=glob.glob(path)
for file in files:
    img = plt.imread(file)


#img = plt.imread(
       '/home/dawid/Pobrane/test_9092015/1.png')

    rad1 = azimuthalAverage(img, center=None,
           stddev=False, returnradii=False,
           return_nr=False,
        binsize=1, weights=None, steps=False,
                interpnan=False, left=None,
                right=None,
        mask=None )
    rad2 = azimuthalAverage(img, center=None,
           stddev=True, returnradii=False,
           return_nr=False,
        binsize=1, weights=None, steps=False,
                interpnan=False, left=None,
                right=None,
        mask=None )


#import numpy as np

#import matplotlib.pyplot as plt

#from scipy.optimize import curve_fit
#def fitFunc(t, a, b, c):
 #   return a*sin(b+t) + c
#N = 400 # number of data points
#p = np.linspace(0, 400, N)
#temp = fitFunc(rad1, 1.5, 1.3, 1.5)
#y = np.array(rad1)
#fitParams, fitCovariances = curve_fit(fitFunc,
                                      t, y)
#print fitParams
#print fitCovariances
#plt.ylabel('Temperature (C)', fontsize = 16)
#plt.xlabel('time (s)', fontsize = 16)
#plt.xlim(0,50.1)
# plot the data as red circles with errorbars
in the vertical direction
#plt.errorbar(t, y, fmt = 'ro', yerr = 0.4)
# now plot the best fit curve and also +- 3
sigma curves
# the square root of the diagonal covariance
matrix element
# is the uncertianty on the corresponding fit
parameter.
#sigma = [fitCovariances[0,0], fitCovariances[1,
          1], fitCovariances[2,2] ]
#plt.plot(t, fitFunc(t, fitParams[0], fitParams[
           1], fitParams[2]),\
#t, fitFunc(t, fitParams[0] + sigma[0],
             fitParams[1] - sigma[1],
```

```
#fitParams[2] + sigma[2]),\
#t, fitFunc(t, fitParams[0] - sigma[0],
            fitParams[1] + sigma[1],
#fitParams[2] - sigma[2])\
#)

    rad3=rad1+rad2
    import matplotlib.pyplot as plt
    x = np.linspace(0, 400, 400)
    y = rad1
    y2=rad3
    z = np.polyfit(x, y, 75)
    p = np.poly1d(z)
    z2 = np.polyfit(x, y2, 75)
    p2 = np.poly1d(z2)
    xp = np.linspace(0, 400, 400)
# save plot to a file
    def movingaverage(interval, window_size):
        window= np.ones(int(window_size))/float(
                window_size)
        return np.convolve(interval, window,
                           'same')
    y_av = movingaverage(rad1, 10)
    y_av2 = movingaverage(rad3, 10)
    max_y = max(y_av)  # Find the maximum y
            value
    max_x = x[y_av.argmax()]  # Find the x
            value corresponding to the maximum
            y value
#print max_x, max_y
    max_y2 = max(y_av2)  # Find the maximum y
             value
    max_x2 = x[y_av2.argmax()]
##########################################
##################################
```

- Własny algorytm - znane parametry i wzory;
- Własny algorytm - własne błędy i odpowiedzialność;
- Własne oprogramowanie - pełna automatyzacja pomiaru.

Dziękuję za uwagę