Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ _____

КАФЕДРА _____

Лабораторная работа №6

Студент ____РЛ2-85_____ _____ _Ситников Д.С._____
(Группа) (Подпись, дата)
(И.О.Фамилия)

Преподователь _____ _Дружин В.В._____
(Подпись, дата) (И.О.Фамилия)

2020 г.

Теоретическая часть

Цель лабораторной работы – практическое освоение интерферометрического метода измерение погрешностей формы сферических поверхностей оптических деталей.

Для контроля формы оптических поверхностей в промышленности широко применяют интерферометрический метод, который реализуют с помощью пробных стекол или интерферометров. Интерферометры позволяют реализовать идею «пробное стекло на расстоянии». Они обеспечивают возможность контролировать форму вогнутых сферических поверхностей большого размера с помощью эталона малого диаметра, а также регистрировать интерференционную картину и извлекать из нее измерительную информацию в реальном масштабе времени.

В данной лабораторной работе для контроля формы вогнутых сферических поверхностей оптических деталей используется интерферометр с квазиапланатическим мениском. Схема интерферометра показана на рис. 1. Источником излучения здесь служит *He-Ne* лазер 1, излучающий на длине волны $\lambda = 0{,}6328$ мкм и обладающий большой длиной когерентности Фокусирующая система 2 создает сферический волновой фронт (гомоцентрический пучок лучей с вершиной в точке *A*). Здесь установлена точечная диафрагма 3, предназначенная для диафрагмирования паразитных световых пучков, которые могут образоваться в результате многократных отражений света на поверхностях линз фокусирующей системы и создать нежелательный фон в плоскости регистрации интерференционной картины. Диаметр точечной диафрагмы обычно немного больше диаметра пятна рассеяния, создаваемого фокусирующей системой 2. Далее излучение проходит через светоделительную куб-призму 4, гипотенузная грань которой служит полупрозрачным зеркалом, отклоняющем лучи в сторону менисковой линзы 5. Выпуклая сферическая поверхность менисковой линзы расположена таким образом, что лучи падают на нее по нормалям. При взаимодействии с выпуклой поверхностью линзы 5 световой пучок по интенсивности разделяется на два пучка: один пучок автоколлимирует от нее, а второй проходит к контролируемой детали 6.

Отраженные от выпуклой поверхности Э менисковой линзы лучи формируют эталонный сферический волновой фронт, поэтому эту поверхность называют эталонной поверхностью интерферометра. Прошедший к контролируемой поверхности К пучок называют рабочим пучком. Лучи рабочего пучка суть нормали к поверхности К, поэтому после отражения они автоколлимируют, формируя рабочий волновой фронт, искаженный погрешностями формы поверхности К. В обратном ходе эталонный и рабочий волновые фронты взаимодействуют между собой, образуя рабочую интерференционную картину, в которой содержится информация о

погрешностях формы поверхности К. Вследствие большой длины когерентности лазерного излучения интерференционная картина одинаково контрастна в любой плоскости, где существуют интерферирующие пучки.

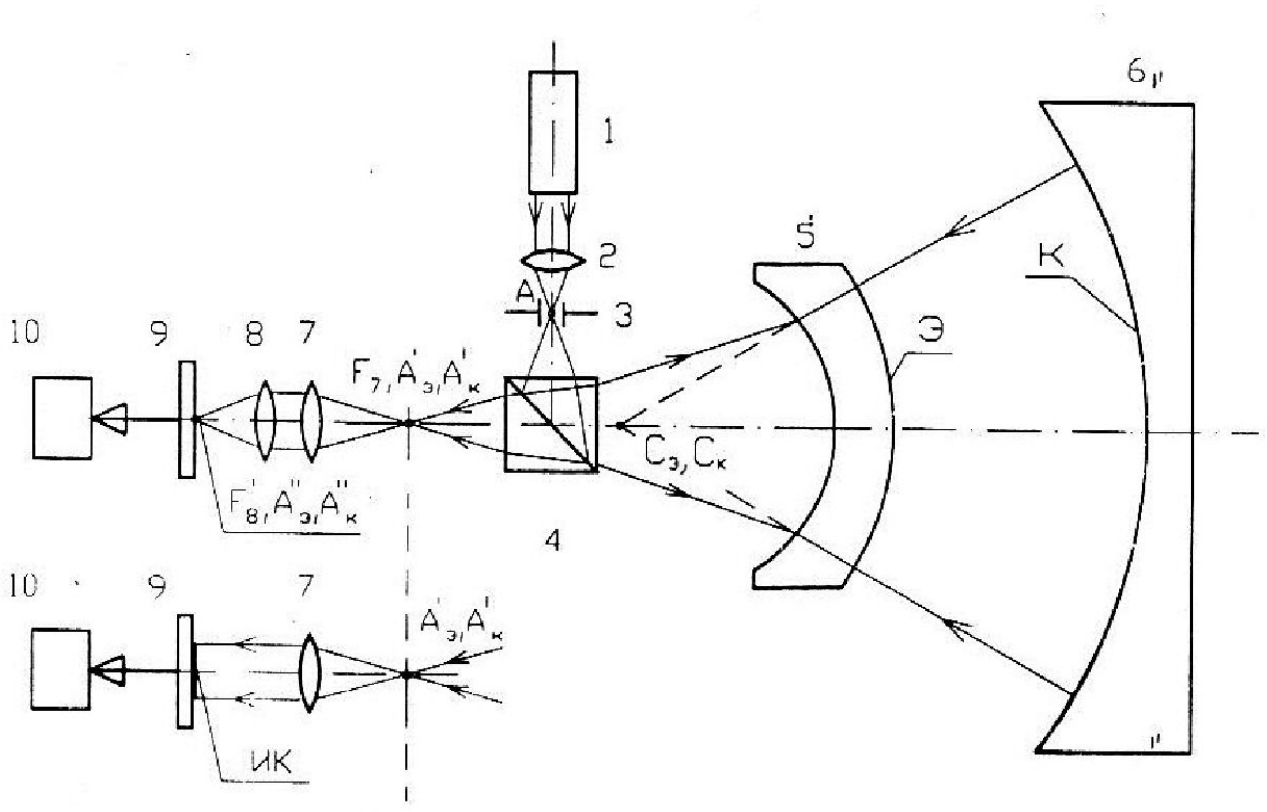


Рис. 1.

Регистрация интерференционной картины осуществляется фотоэлектрическим способом, для чего используются координатный приемник излучения 7 и персональный компьютер 8, обеспечивающий запись интерферограммы в файл и последующую ее обработку с целью извлечения измерительной информации. Регистрирующая ветвь интерферометра устроена таким образом, чтобы попеременно можно было наблюдать автоколлимационные точки или рабочую интерференционную картину.

Чтобы установить связь между рассогласованием центров кривизны волновых фронтов, обратимся к рис. 2. Здесь показаны различные варианты взаимного расположения двух сферических волновых фронтов W1 и W2. В плоскости анализа волновые фронты характеризуются радиусами кривизны R1 = R2. С учетом приблизительного равенства в плоскости анализа значений радиусов кривизны интерферирующих волновых фронтов, обозначим радиусы фронтов буквой R.

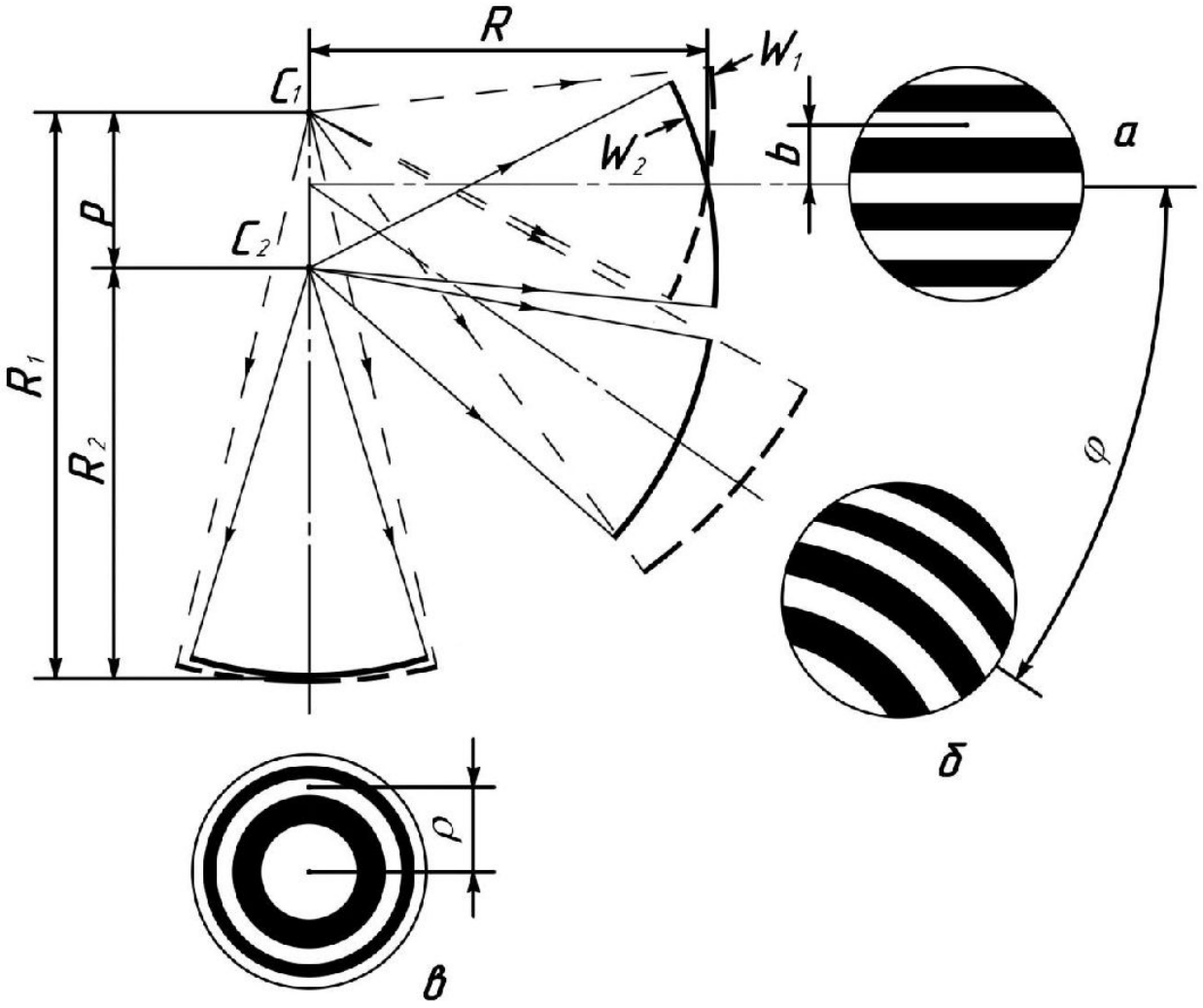


Рис. 2.

В случае рис. 2, а центры С1 и С2 волновых фронтов разведены на расстояние p в плоскости, перпендикулярной направлению их распространения, причем расстояние p << R . Тогда угол ω между интерферирующими лучами по всему полю интерференции будет практически постоянным

$$\omega = \frac{p}{R}.$$

Ширину b полос в этом случае вычисляют по формуле

$$b = \frac{\lambda}{\omega} = \frac{\lambda R}{p}$$

из которой следует, что полосы на интерференционной картине будут прямые и одинаковой ширины. Очевидно, что при диаметре DИ интерферограммы мы будем наблюдать $N_{\text{полос}}$ полос, определяемое из соотношения

$$N_{\text{полос}} = \frac{D_и}{b} = \frac{D_и p}{\lambda R} - 2p * \frac{sin\sigma}{\lambda},$$

где $sin\sigma = D_и/2R_1$ числовая апертура интерферирующих пучков.

В случае рис. 2, в направление распространения волновых фронтов совпадает с линией, проходящей через точки С1 и С2. Здесь до поля анализа доходят волновые фронты, имеющие различные радиусы R1 = R2, причем R1 = R2 + p. Интерференционная картина этом случае будет иметь вид

концентрических колец, называемых кольцами Ньютона. Радиус ρ_m светлого кольца, имеющего порядок интерференции (m = 0; 1; 2; 3 и т.д.), определяют по формуле, полученной Ньютоном

$$\rho_m = \frac{\sqrt{2m\lambda R_2^2}}{p},$$

из которой следует, что ширина колец убывает от центра к периферии. Число $N_{\text{колец}}$ светлых колец равно порядку интерференции крайнего кольца и определяется из формулы

$$N_{\text{колец}} = \frac{D_и p}{8\lambda R^2} = p\frac{sin^2\sigma}{2\lambda}.$$

В случае, представленном на рис. 2, б, направление распространения интерферирующих волн таково, что оно составляет угол φ по отношению к случаю рис. 2, а и угол $(90° - \varphi)$ по отношению к линии, соединяющей точки C1 и C2. Поэтому здесь центры волновых фронтов одновременно рассогласованы в двух взаимно-перпендикулярных направлениях и интерференционная картина здесь имеет вид колец Ньютона, центр которых смещен в сторону. На рис. 2, б центр колец расположен за пределами поля интерференции. При условии $p << R$ в центре поля интерференции угол ω_φ между интерферирующими лучами определяется соотношением

$$\omega_\varphi = p * \frac{\sin sin\,(90-\varphi)}{R} = \omega\, cos\varphi.$$

Очевидно, что достаточно легко смоделировать вид интерферограмм, полученных при взаимодействии двух строго сферических волновых фронтов. Поэтому при интерферометрических измерениях сравнивают реальную интерферограмму с ее идеальной моделью. Несоответствие между ними выражают в количественной мере. Обычно из интерферограмм извлекают следующую количественную информацию:

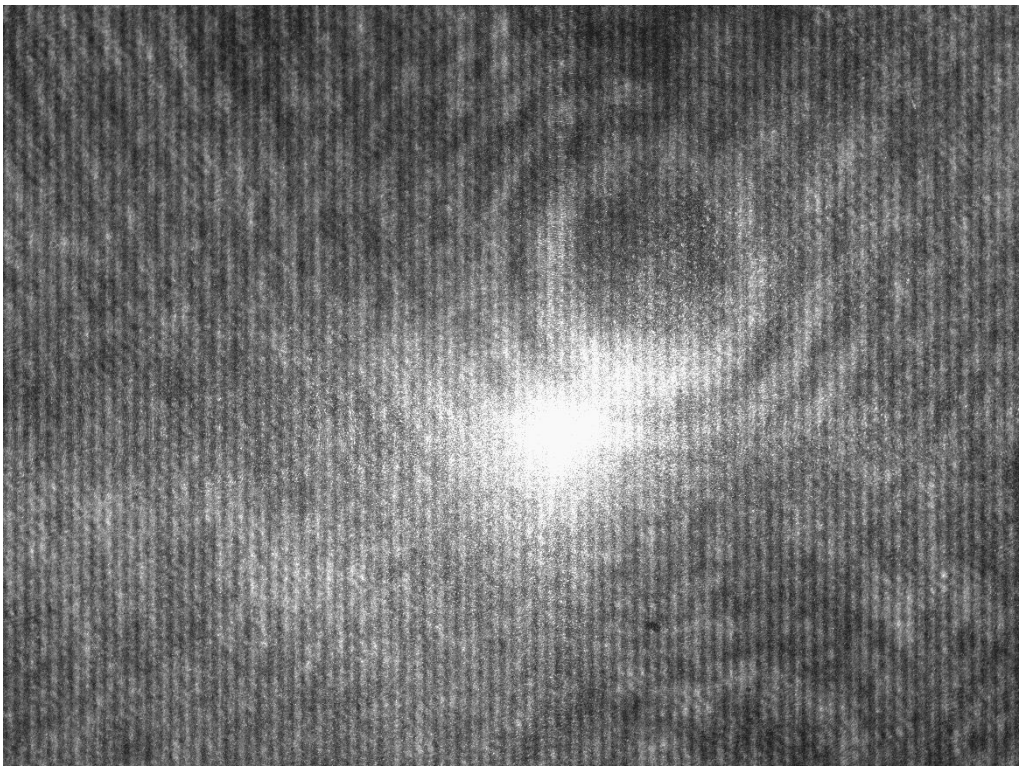
- число интерференционных полос или колец;
- ширину полос или диаметры колец;
- степень и характер искривления полос по отношению к прямой линии;
- степень эллиптичности колец.

Также учитывают направление полос и ориентация осей эллиптических колец.

Так как шириной интерференционной полосы называют интервал между центрами двух соседних одноименных полос, то при подсчете их числа за одну полосу принимают совокупность двух соседних полос: светлой и темной. Аналогичным образом определяют число интерференционных колец. Если число светлых и темных полос или колец одинаково, то это значит, что интерферограмма содержит целое число полос или колец. Когда число светлых полос отличается от числа темных на одну полосу, то общее число полос определяют с точностью до половины полосы.

Практическая часть

Задача - получить по изображению координаты (в пикселях) интерференционных максимумов. Достаточно получить координаты в одном сечении



Будем анализировать сечение с координатой y = 132 (начало координат лежит в левом верхнем угле). Размер изображения составляет 505x372. Для анализа сечения будет использоваться язык программирования python. Для начала импортируем необходимые модули:

- PIL – нужен для обработки графики в Python;
- Matpoltlit необходим для визуализации данных двумерной (2D) графики;
- Numpy – добавляет поддержку больших многомерных массивов и матриц, вместе с большой библиотекой высокоуровневых (и очень быстрых) математических функций для операций с этими массивами.

```
from PIL import Image, ImageDraw
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
from numpy import array, arange
```

Далее получим в заданном сечении распределение цвета в формате (R, G, B) для каждого пикселя.

```
image = Image.open("1.jpg")
draw = ImageDraw.Draw(image)
width = image.size[0]
height = image.size[1]
pix = image.load()
print(width, height)

color = [[pix[j, i][0] for j in range(width)] for i in range(height)]
```

И построим график. График строится с помощью функции printf (см. приложение 1).

```
y = 132 # заданное сечение
printf(color[y], "orig")
```

Так как изображение черно-белое, то значения R=G=B, поэтому построим только одну компоненту

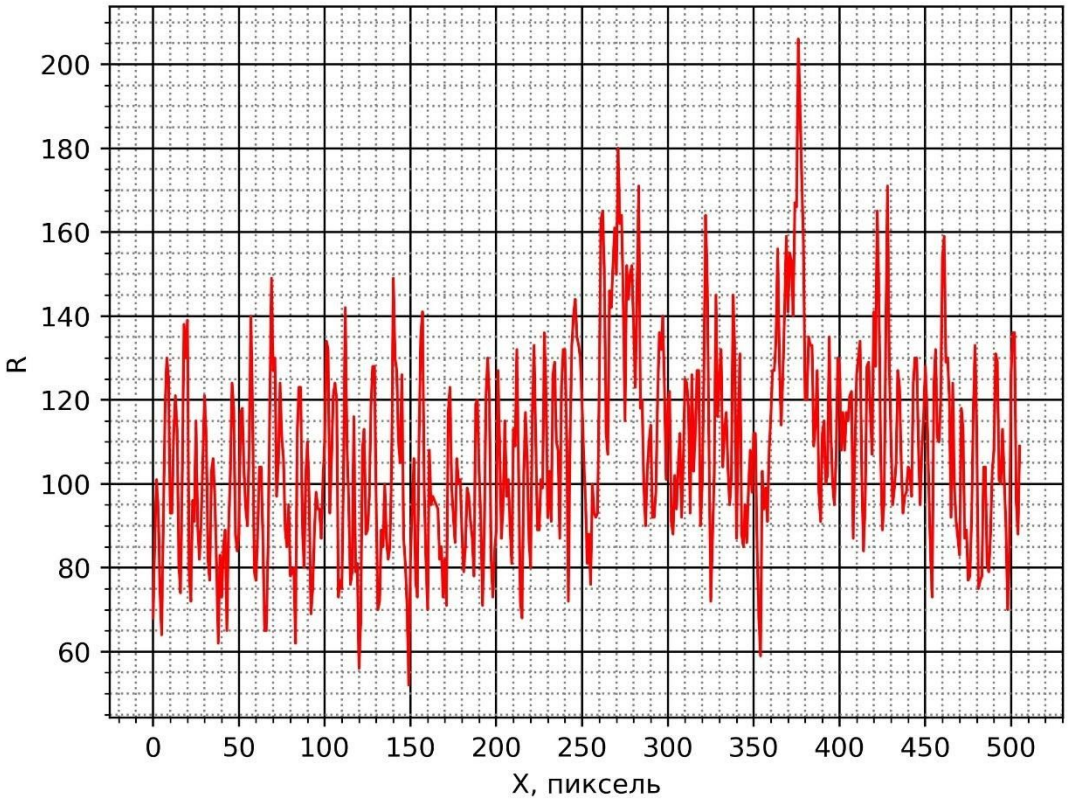


Рис. 3.

Как видно график мало информативен, так как присутствует большое количество шумов. Попробуем избавить от импульсных помех с помощью медианного фильтра. Его суть в том, что он выбирает из группы входных

значений среднее и выдает на выход. Причем обычно группа имеет нечетное количество значений. Возьмем N = 5 и для каждого i пикселя выберем средний в диапазоне $[x_{i-2}; x_{i+2}]$. Для этого обрежем наш массив пикселей на заданный диапазон отсортируем его по порядку и выберем среднее значение. Недостатки метода в том, что краевые значения не фильтруются, но они нас не интересуют, поэтому их отбросим.

```
def median(j, mas):
    mas = mas[j - 2: j + 3]
    mas.sort()
    return mas[2]
```

```
for j in range(2, width - 2):
    color[y][j] = median(j, color[y])
printf(color[y][2:len(color[y]) - 2], "median")
```

Итоговый график

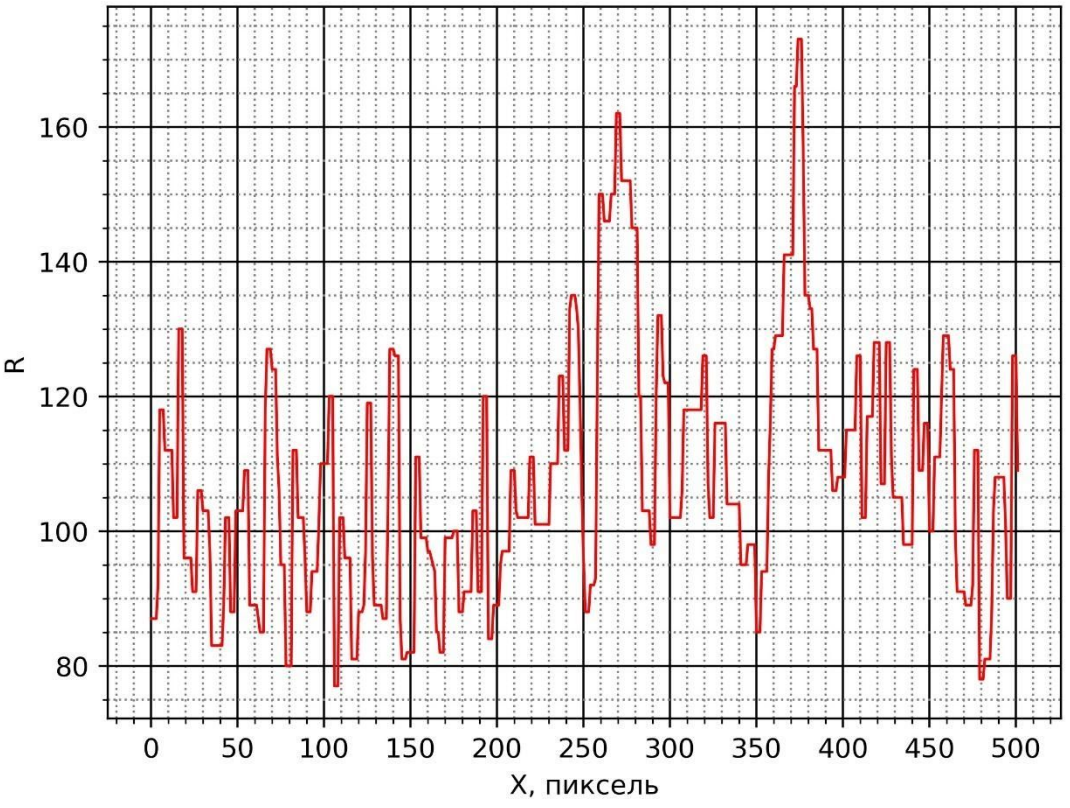


Рис. 4

Разброс значений по амплитуде стал меньше, но анализировать график все еще сложно. Попробуем усреднить сигнал с помощью метода скользящего среднего. Суть метода в том, что для каждого пикселя мы берем сумму значения в диапазоне $[x_i; x_{i+w}]$ и делим на их количество (w). Недостатки метода в том, что при большом w теряется амплитуда сигнала, и также, как в предыдущем методе, краевые значения не сглаживаются.

Таким образом итоговый график

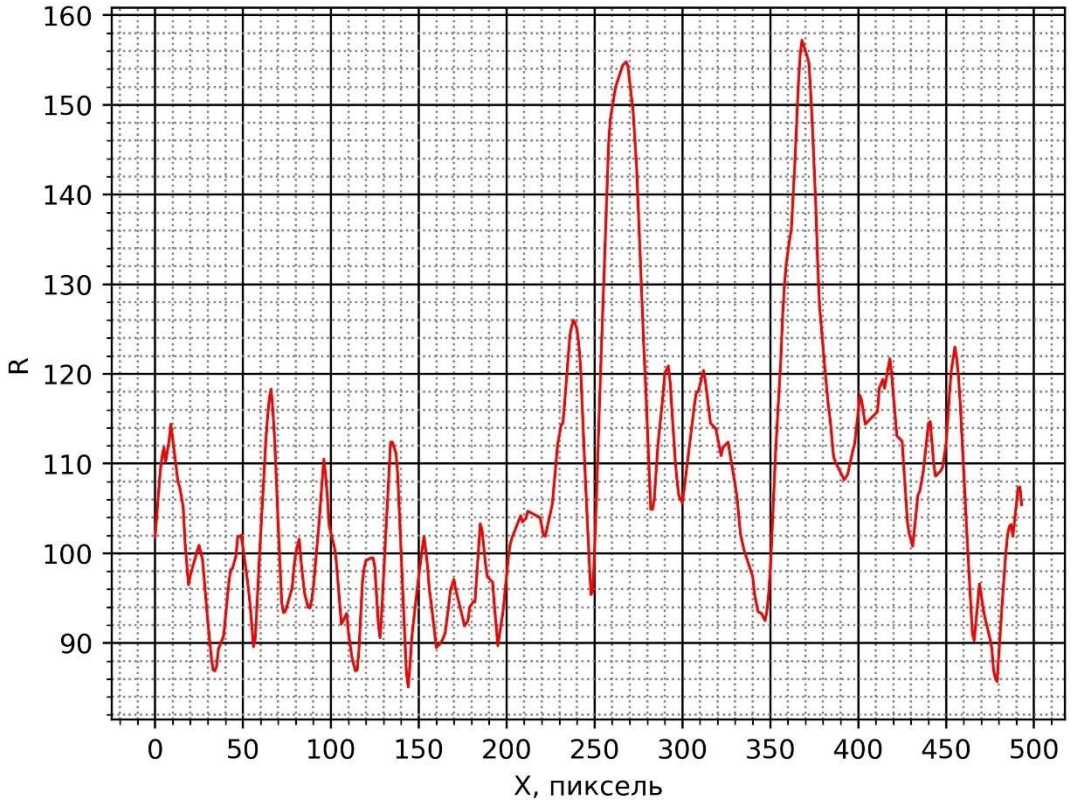


Рис. 5.

Если взглянуть на оригинальное изображение, то можно заметить, что полосы перестают быть видимыми пир x = 130. С помощью графика легко определить координаты интерференционных максимумов

m	x_i	y_i
1	269	132
2	249	132
3	211	132
4	189	132

Приложения

```python
from PIL import Image, ImageDraw
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
from numpy import array, arange

def median(j, mas):
    mas = mas[j - 2: j + 3]
    mas.sort()
    return mas[2]

def printf(mas, filename):
    y = array(mas)
    x = arange(len(mas))
    fig = plt.figure()
    ax = fig.add_subplot(111)
    # fig, ax = plt.subplots()
    ax.plot(x, y, color = 'r', linewidth = 1)
    #  Устанавливаем интервал основных и
    #  вспомогательных делений:
    ax.xaxis.set_major_locator(ticker.MultipleLocator(50))
    ax.xaxis.set_minor_locator(ticker.MultipleLocator(25))
    # ax.yaxis.set_major_locator(ticker.MultipleLocator(0.2))
    # ax.yaxis.set_minor_locator(ticker.MultipleLocator(0.1))
    #  Добавляем линии основной сетки:
    ax.grid(which='major',color = 'k')
    #  Включаем видимость вспомогательных делений:
    ax.minorticks_on()
    #  Теперь можем отдельно задавать внешний вид
    #  вспомогательной сетки:
    ax.grid(which='minor', color = 'gray', linestyle = ':')
    ax.set_xlabel('X, пиксель')
    ax.set_ylabel('R')
    plt.savefig(filename + ".jpeg", format='jpeg', dpi = 400)
    plt.close(fig)
    plt.clf()

```

Рис.1. Код для анализа изображения

```
image = Image.open("1.jpg")
draw = ImageDraw.Draw(image)
width = image.size[0]
height = image.size[1]
pix = image.load()

color = [[pix[j, i][0] for j in range(width)] for i in range(height)]

y = 132 # заданное сечение
printf(color[y], "orig")

k = 1

for j in range(2, width - 2):
    color[y][j] = median(j, color[y])
printf(color[y][2:len(color[y]) - 2], "median")

w = 10
for j in range(width):
    sum1 = 0
    if (j + w > width):
        break
    for h in range(j, j + w):
        sum1 += color[y][h]
    color[y][j] = sum1 / w

printf(color[y][130:len(color[y]) - w], "sred")
```

Рис. 2. Код для анализа изображения