МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра Вычислительной техники

Математические методы распознавания образов.

Лабораторная работа №2

Калибровка камеры с помощью Python - OpenCV

|  |  |
| --- | --- |
| Факультет: АВТФ  Группа: АПИМ-24  Выполнил:  Разуваев В. В. | Проверил:  Ильиных С.П. |

**Содержание**

[Цель работы 3](#_Toc193843402)

[Задание 3](#_Toc193843403)

[Ход работы 3](#_Toc193843404)

[Задача 1 3](#_Toc193843405)

[Задача 2-3 6](#_Toc193843406)

# Цель работы

Освоить базовые навыки калибровки камеры с использованием Python - OpenCV.

# Задание

1. Выполнить калибровку камеры с использованием шахматной доски.
2. Используя найденные параметры, следует исправить исходного изображения.
3. Получить исправленное изображение путем репроектирования.

# Ход работы

## Задача 1

*Листинг 1 – Код для получения параметров камеры*

# Import required modules

import cv2

import numpy as np

import os

import glob

# Define the dimensions of checkerboard

CHECKERBOARD = (5, 7)

# stop the iteration when specified

# accuracy, epsilon, is reached or

# specified number of iterations are completed.

criteria = (cv2.TERM\_CRITERIA\_EPS + cv2.TERM\_CRITERIA\_MAX\_ITER, 30, 0.001)

# Vector for 3D points

threedpoints = []

# Vector for 2D points

twodpoints = []

# 3D points real world coordinates

objectp3d = np.zeros((1, CHECKERBOARD[0] \* CHECKERBOARD[1], 3), np.float32)

objectp3d[0, :, :2] = np.mgrid[0:CHECKERBOARD[0], 0:CHECKERBOARD[1]].T.reshape(-1, 2)

prev\_img\_shape = None

# Extracting path of individual image stored

# in a given directory. Since no path is

# specified, it will take current directory

# jpg files alone

images = glob.glob("\*.jpg")

for filename in images:

    image = cv2.imread(filename)

    grayColor = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

    # Find the chess board corners

    # If desired number of corners are

    # found in the image then ret = true

ret, corners = cv2.findChessboardCorners(grayColor, CHECKERBOARD, cv2.CALIB\_CB\_ADAPTIVE\_THRESH + cv2.CALIB\_CB\_FAST\_CHECK + cv2.CALIB\_CB\_NORMALIZE\_IMAGE)

# If desired number of corners can be detected then,

# refine the pixel coordinates and display

# them on the images of checker board

if ret == True:

    threedpoints.append(objectp3d)

    # Refining pixel coordinates

    # for given 2d points.

    corners2 = cv2.cornerSubPix(grayColor, corners, (11, 11), (-1, -1), criteria)

    twodpoints.append(corners2)

    # Draw and display the corners

    image = cv2.drawChessboardCorners(image, CHECKERBOARD, corners2, ret)

    cv2.imshow("img", image)

    cv2.waitKey(0)

cv2.destroyAllWindows()

h, w = image.shape[:2]

# Perform camera calibration by

# passing the value of above found out 3D points (threedpoints)

# and its corresponding pixel coordinates of the

# detected corners (twodpoints)

ret, matrix, distortion, r\_vecs, t\_vecs = cv2.calibrateCamera(

threedpoints, twodpoints, grayColor.shape[::-1], None, None)

# Displayig required output

print(" Camera matrix:")

print(matrix)

print(" Distortion coefficient:")

print(distortion)

print(" Rotation Vectors:")

print(r\_vecs)

print(" Translation Vectors:")

print(t\_vecs)

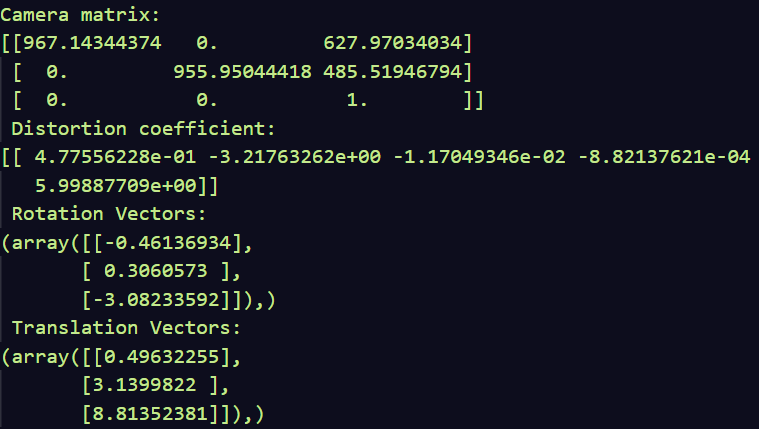
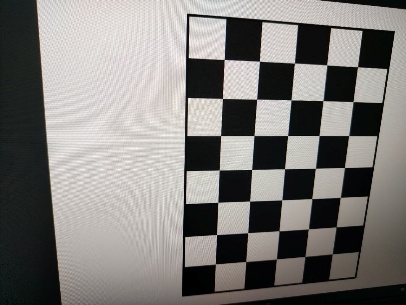
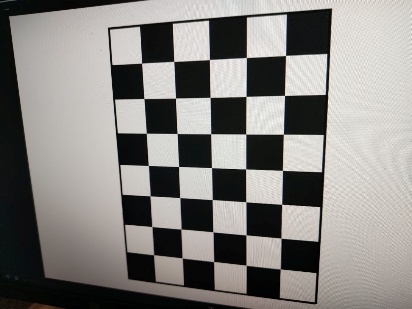
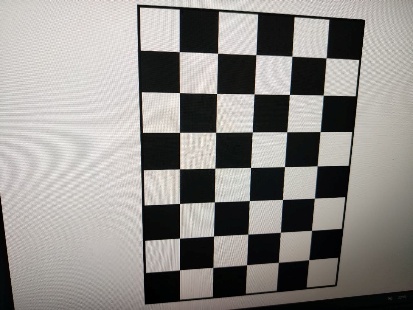


Рисунок 1 – Результат работы программы

Для калибровки было использовано 5 изображений шахматной доски размером 5 на 7 углов.



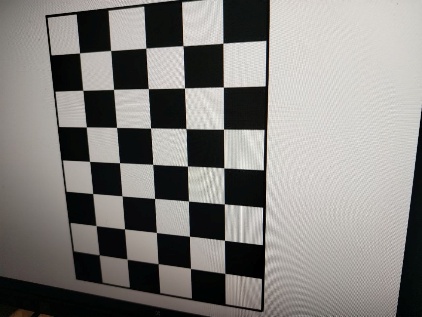
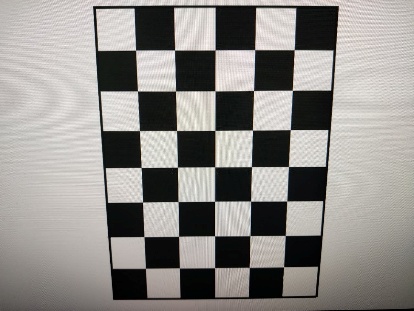


Рисунок 2 – Калибровачные изображения

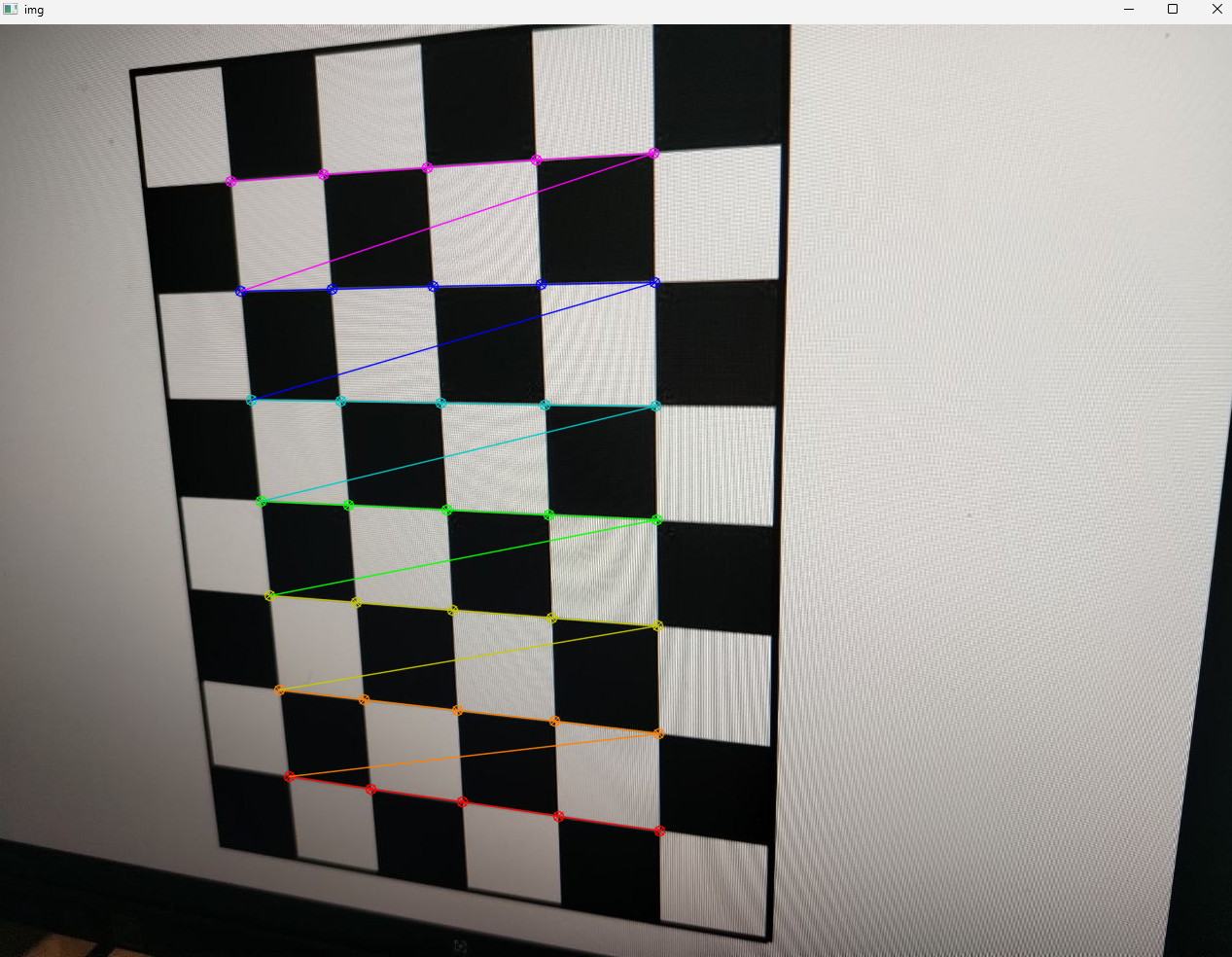


Рисунок 3 – Отображения обнаруженных углов шахматной доски

## Задача 2-3

*Листинг 2 – Код для исправления изображения*

import cv2

import numpy as np

# Задаём параметры калибровки, полученные ранее:

camera\_matrix = np.array([[967.14344374,   0.        , 627.97034034],

                          [  0.        , 955.95044418, 485.51946794],

                          [  0.        ,   0.        ,   1.        ]])

dist\_coeffs = np.array([[ 4.77556228e-01, -3.21763262e+00, -1.17049346e-02, -8.82137621e-04,

                          5.99887709e+00]])

# Загружаем исходное изображение с дисторсией

img = cv2.imread("5.jpg")  # замените "your\_image.jpg" на имя вашего файла

h, w = img.shape[:2]

# Получаем оптимизированную новую матрицу камеры,

# которая может учитывать интересующую нас область изображения.

# new\_camera\_matrix, roi = cv2.getOptimalNewCameraMatrix(camera\_matrix, dist\_coeffs, (w, h), 1, (w, h))

# Получаем новую матрицу камеры с alpha=1 (сохраняем полный кадр)

new\_camera\_matrix, roi = cv2.getOptimalNewCameraMatrix(camera\_matrix, dist\_coeffs, (w, h), alpha=0.7, newImgSize=(w, h))

# Инициализируем карты для функции cv2.remap

mapx, mapy = cv2.initUndistortRectifyMap(camera\_matrix, dist\_coeffs, None, new\_camera\_matrix, (w, h), cv2.CV\_32FC1)

# Выполняем репроекцию (исправление дисторсии)

# undistorted\_img = cv2.remap(img, mapx, mapy, cv2.INTER\_LINEAR)

# Затем выполняем remap без обрезки по ROI:

undistorted\_img = cv2.remap(img, mapx, mapy, cv2.INTER\_LINEAR)

# (Необязательно) Обрезаем изображение по ROI, если нужно

# x, y, w\_roi, h\_roi = roi

# undistorted\_img = undistorted\_img[y:y+h\_roi, x:x+w\_roi]

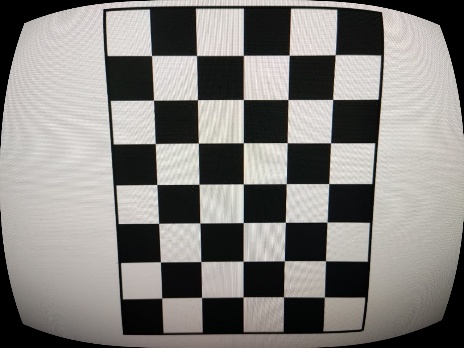
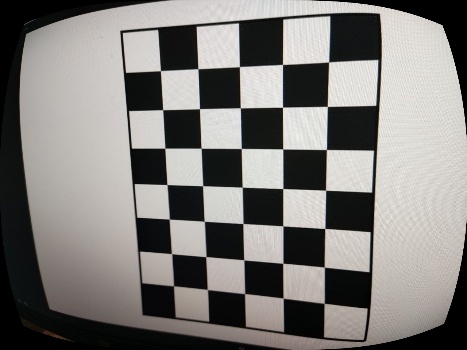
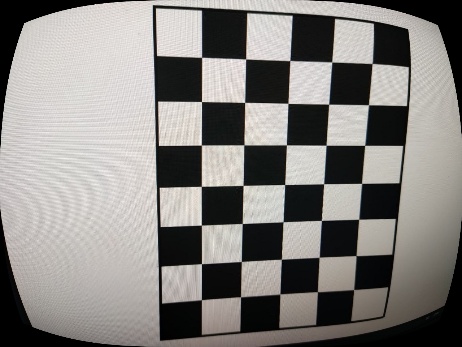
# Сохраняем и отображаем результат

cv2.imwrite("undistorted\_image15.jpg", undistorted\_img)

cv2.imshow("Undistorted Image", undistorted\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()



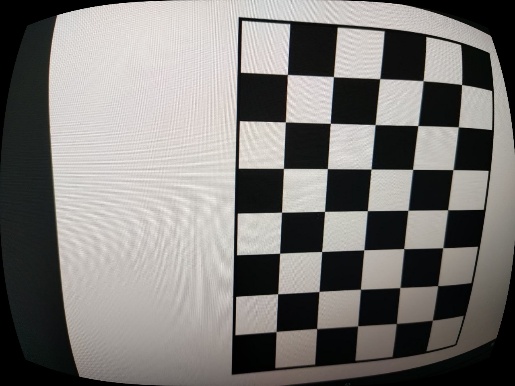
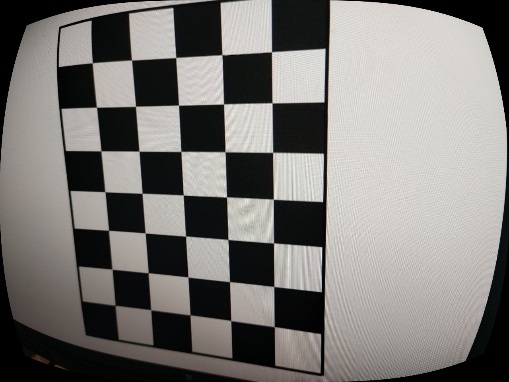


Рисунок 4 – Исправленные изображения