ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЩОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«**САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО**»
Институт машиностроения, материалов и транспорта
Высшая школа автоматизации и робототехники

# Отчет о прохождении производственной практики

По направлению подготовки: 15.03.06 – «Мехатроника и робототехника»
Место прохождения практики: «Центральный научно-исследовательский и опытно-конструкторский институт робототехники и технической кибернетики»
Сроки практики: 10.06.2023 – 08.07.2023

| | | |
|---|---|---|
| Руководитель практики: | | Уланов В. Н. |
| Научный руководитель: | Оценка:____ | Громошинский Д. А. |
| Выполнил студент гр. 3331506/00401 | | Поздняков С. А. |
| Дата:_____ | | |

Санкт-Петербург
2023

# Оглавление

# Введение

Компьютерное зрение – это одна из самых востребованных областей на современном этапе развития цифровых компьютерных технологий. Оно требуется на производстве, при управлении роботами, при автоматизации процессов, в медицинских и военных приложениях, при наблюдении со спутников и при работе с персональными компьютерами, в частности поиске цифровых изображений.[1] Системы технического зрения (СТЗ) призваны и во многих случаях уже решают задачи по дополнению и даже замене человека в областях деятельности, связанных со сбором и анализом зрительной информации. Уровень их использования в прикладных областях является одним из наиболее наглядных показателей уровня развития высоких технологий. Цель компьютерного зрения заключается в формировании полезных выводов относительно объектов и сцен реального мира на основе анализа изображений, полученных с помощью датчиков (камер).[2]

## Задание

*Построение трехмерной поверхности рабочей зоны манипулятора по двум кадрам с видеокамеры робота при известном начальном приближении положения камеры относительно базы робота.*

Для реализации поставленной задачи было необходимо получить основные навыки работы с операционной системой Linux (Ubuntu 20.04) и особенностями взаимодействия с данной системой, изучить библиотеку OpenCV. OpenCV (Opensource computer vision library) – библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом. Для выполнения данного задания был использован язык программирования C++, но также возможно использование данной библиотеки с такими языками программирования, как Python, Java, MATLAB со своими особенностями для каждого.

Для работы была установлена библиотека OpenCV 4.6.0 и использовались основные её модули:

- opencv_core – основные структуры, вычисления, математические функции, линейная алгебра, ввод/вывод;
- opencv_highgui – ввод/вывод изображений и видео;
- opencv_video – анализ движения и отслеживания объектов;
- opencv_imgproc – обработка изображений (фильтрация, геометрические преобразования, преобразование цветовых пространств);
- opencv_objdetect – обнаружение объектов на изображении.

## Основные возможности библиотеки OpenCV

### Чтение изображения из файла

Чтение изображения из файла осуществляется по его полному адресу (расположению). Применение такого алгоритма возможно для поиска изображений с особыми необходимыми параметрами, поскольку адрес изображения может быть представлен переменной и изменять свое значение, в связи с чем будет меняться обрабатываемое изображение. Имя задается переменной типа string, но при использовании нумерации изображений, формирующих необходимый набор исследуемых материалов возможно обращение к последующему изображению путем инкрементирования адреса и перевода из типа данных unsigned int в string. Пример программы, реализующий данный метод, и результат её работы представлены на рисунке 1 и рисунке 2.

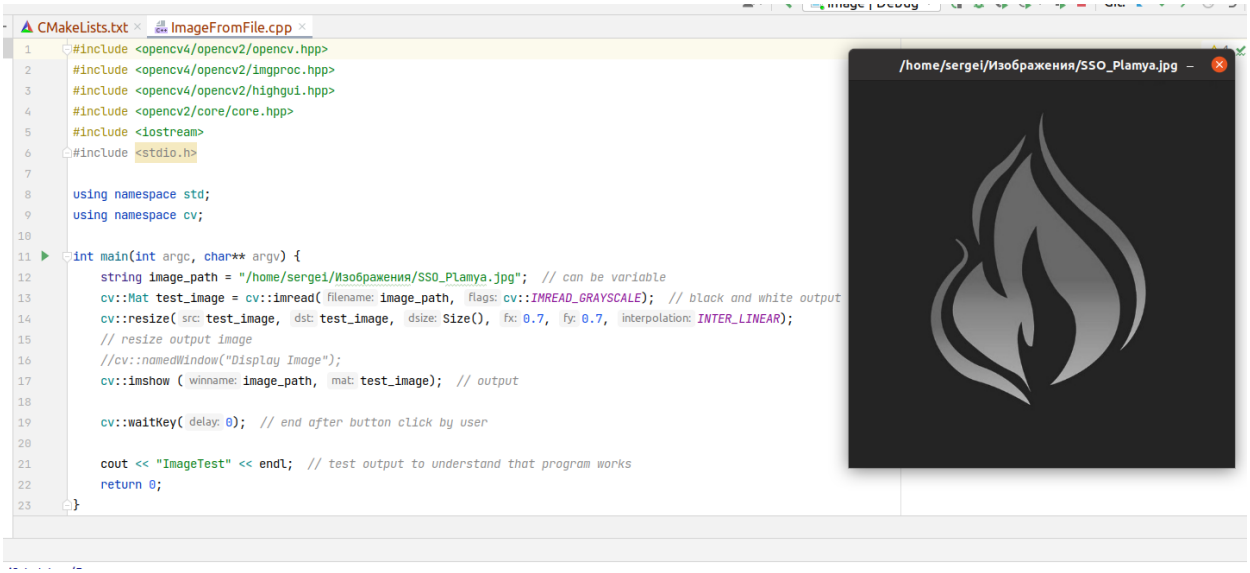


Рисунок 1

```
cmake_minimum_required(VERSION 3.25)
project(ImageFromFile)
find_package( OpenCV REQUIRED )
include_directories( ${OpenCV_INCLUDE_DIRS} )
set(CMAKE_CXX_STANDARD 17)
add_executable(ImageFromFile ImageFromFile.cpp)
target_link_libraries( ImageFromFile ${OpenCV_LIBS} )
```

Рисунок 2 — CMake file

Чтение видеозаписи из файла

Чтение видеозаписи из файла осуществляется по его полному адресу и возможно для детектирования объектов. Данный способ может быть применен для набора видеозаписей, поскольку имя может быть задано переменной типа string. Пример программы, реализующей такую функцию, и результат её работы представлены на рисунке 3 и рисунке 4.

CMakeLists.txt | Video.cpp

```cpp
#include <opencv4/opencv2/opencv.hpp>
#include <opencv4/opencv2/imgproc.hpp>
#include <opencv4/opencv2/highgui.hpp>
#include <opencv4/opencv2/core/core.hpp>
#include <iostream>
#include <stdio.h>

using namespace std;
using namespace cv;

int main(int argc, char** argv) {
    Mat video;
    VideoCapture capture;
    string filename = "/home/sergei/Видео/BOATS.MOV";
    capture.open(filename);

    while (1) {
        capture >> video;

        if (video.empty()) break;

        resize( src: video, dst: video, dsize: Size(), fx: 0.55, fy: 0.55, interpolation: INTER_NEAREST);
        // resize output video
        imshow( winname: filename, mat: video);
        // output video on screen
        if (waitKey( delay: 30) >= 0) break;
        //click escape button to turn video off
    }
    cout << filename << endl;
    return 0;
}
```


Рисунок 3

```
cmake_minimum_required(VERSION 3.25)
project( VideoFromFile )
find_package( OpenCV REQUIRED )
include_directories( ${OpenCV_INCLUDE_DIRS} )
add_executable( Video Video.cpp)
target_link_libraries( Video ${OpenCV_LIBS} )
```

Рисунок 4 — CMake file

Вывод видеоизображения с подключенной видеокамеры

Получение видеоизображения производится с помощью команды VideoCapture capture.open(idx) по индексу камеры, подключенной к компьютеру. Для встроенное видеокамеры будет idx = 0, для сторонних, подключенных через имеющиеся интерфейсы – «1», «2» и т. д. Данный способ может бы применен для детектирования требуемых объектов в режиме реального времени (например наличие средств индивидуальной защиты у работников на строительном объекте или определение номера автомобиля). Пример программы, реализующей такую функцию, и результат её работы представлены на рисунке 5 и рисунке 6.

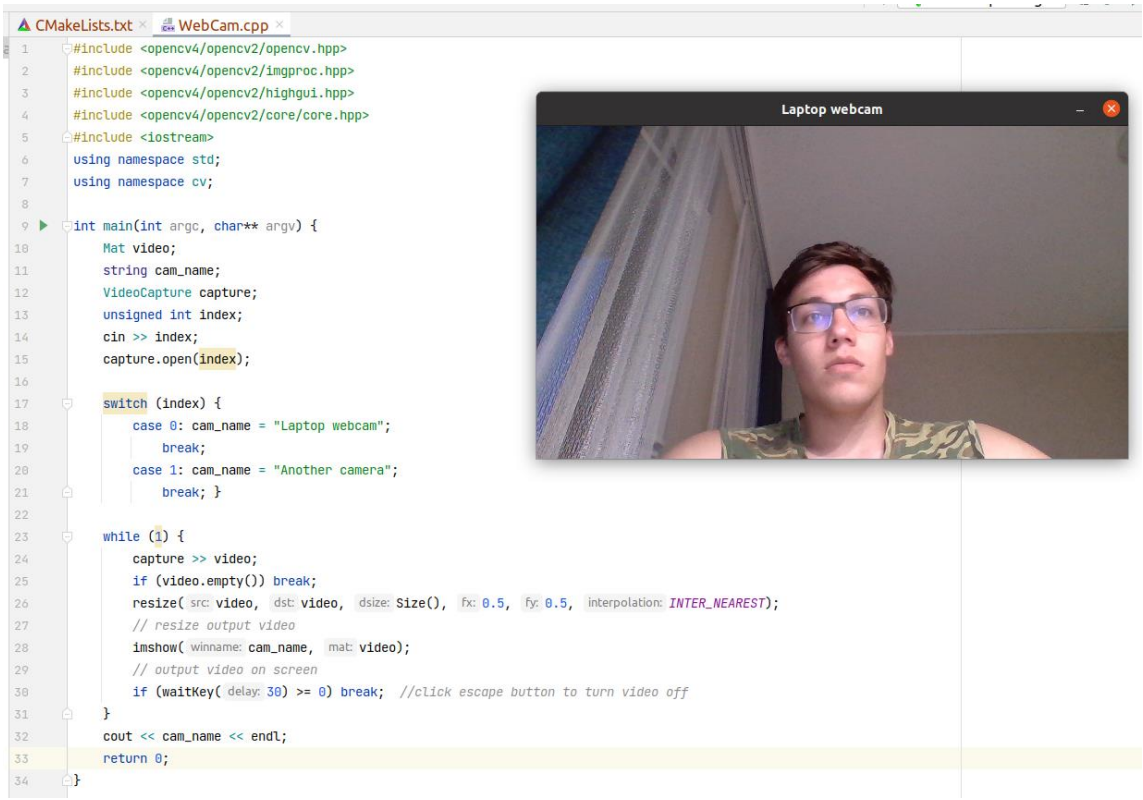


Рисунок 5

```
cmake_minimum_required(VERSION 3.25)
project(WebVideo)
add_executable(WebVideo WebVideo.cpp)
set(CMAKE_CXX_STANDARD 17)
find_package( OpenCV REQUIRED )
include_directories( ${OpenCV_INCLUDE_DIRS} )
target_link_libraries( WebVideo ${OpenCV_LIBS} )
```

Рисунок 6 — CMake file

## Обработка данных с камеры в OpenCV

### Геометрия формирования изображения

Перед началом работы с изображениями, полученными с помощью камеры, следует изучить способ получения изображения и перехода из глобальной системы координат в систему координат камеры для определения координат искомых точек. Рассмотрим глобальную систему координат, связанную с тремя измерениями комнаты, и локальную систему координат камеры.

Задача: учитывая трехмерную точку **P** в комнате необходимо найти координаты пикселей $(u,\ v)$ данной трехмерной точки на двумерном изображении, полученном камерой.

Глобальная система координат

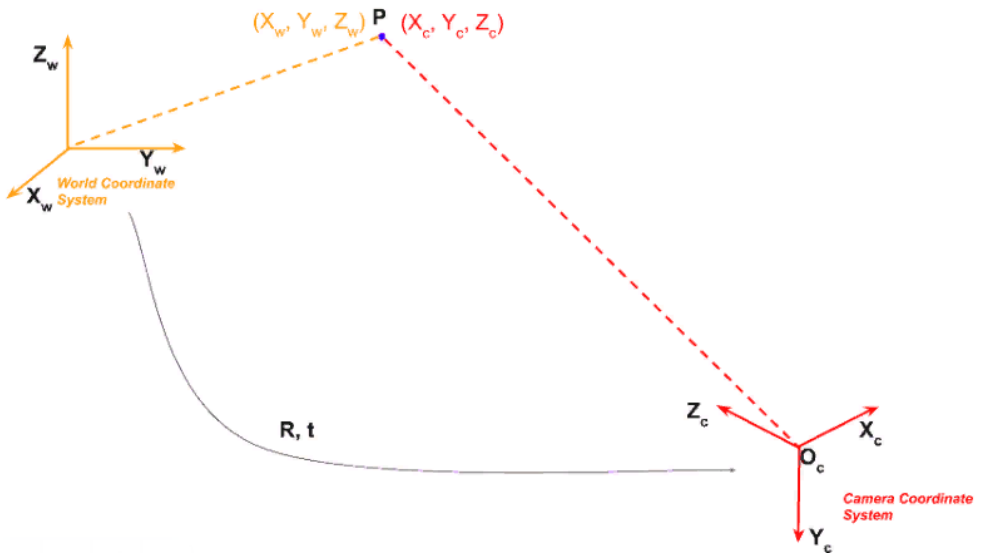


Рисунок 7 – Связь глобальной и локальной систем координат

Глобальная система координат и система координат камеры связаны поворотом и переносом. За данные преобразования отвечают 3 параметра для поворота и 3 для переноса. Данные параметры называются внешними параметрами камеры. Начало координат глобальной системы задается точкой $(0,\ 0,\ 0)$, например в углу комнаты, а оси проходят по ребрам правильного параллелепипеда. Трехмерные координаты каждой точки пространства могут быть заданы тремя координатами в глобальной системе координат, назовем их $(X_w, Y_w, Z_w)$.

Локальная система координат камеры

Изображение в комнате получено с помощью камеры, расположенной в произвольной точке данного пространства, поэтому для определения координат точки на снимке требуется знать координаты камеры (центр локальной системы координат относительно глобальной). Пусть камера расположена в точке $(t_w, t_w, t_w)$. Помимо переноса локальная система координат может быть повернута относительно глобальной. Для математического представления локальной системы координат поворот представляется матрицей вращения 3x3. Глобальная и локальная системы координат связаны матрицей вращения **R** и вектором переноса **t**. Следовательно если точка имела координаты $(X_w, Y_w, Z_w)$, то в системе камеры будет иметь координаты $(X_c, Y_c, Z_c)$. Данные значения координат связаны следующим соотношением:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \boldsymbol{R} \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + \boldsymbol{t},$$

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = [\boldsymbol{R}|\boldsymbol{t}] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}.$$

Вектор переноса и матрица поворота формируют внешнюю матрицу **P**:

$$\mathbf{P} = [\boldsymbol{R}|\boldsymbol{t}].$$

Система координат изображения

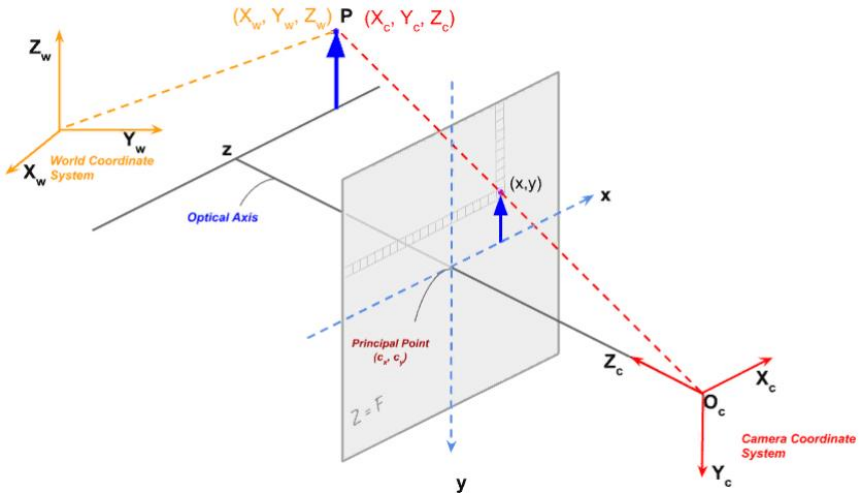


Рисунок 8 – Проекция точки P на плоскость изображения

После получения координат точки в локальной системе камеры необходимо спроецировать искомую точку на плоскость изображения, чтобы получить местоположение точки на изображении. Плоскость изображения расположена на фокусном расстоянии $f$ от оптического центра камеры. Из подобия треугольников нетрудно заметить, что проекционное изображение $(x, y)$ трехмерной точки $(X_c, Y_c, Z_c)$ получается следующим образом:

$$x = f\frac{X_c}{Z_c}, y = f\frac{Y_c}{Z_c}.$$

Ниже представлены данные уравнения в матричной форме:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \boldsymbol{K}\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix},$$

$$\boldsymbol{K} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} - \text{втроенная матрица содержит}$$

внутренние параметры камеры.

Поскольку оптический центр может не совпадать с центром локальной системы координат, а фокусные расстояния по осям могут отличаться, то следует внести изменения в запись матрицы $\boldsymbol{K}$:

$$\boldsymbol{K} = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}.$$

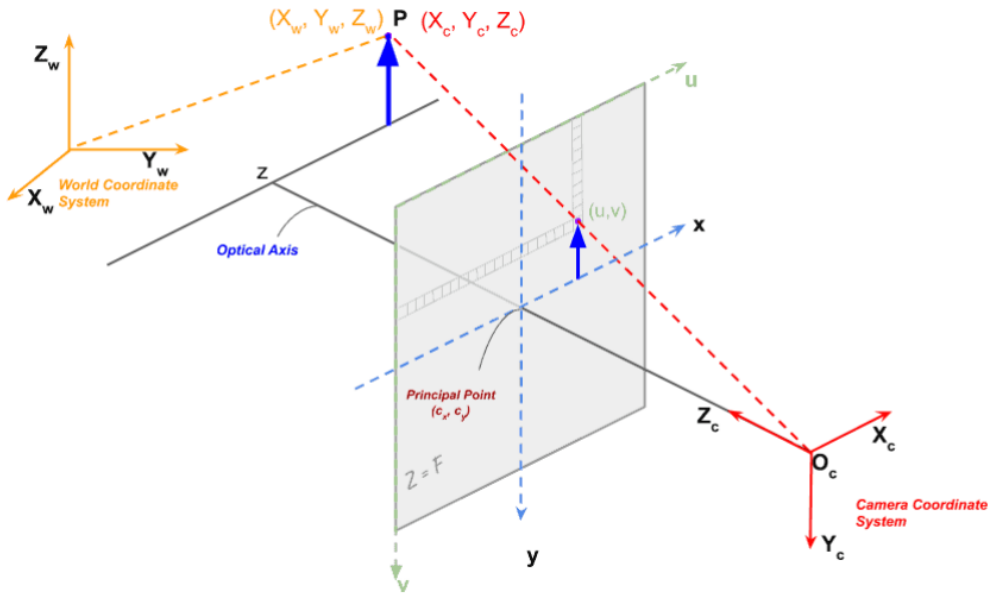


Рисунок 9 – Центр двумерной системы координат изображения не лежит на оптической оси камеры

В приведенном выше уравнении координаты точки на изображении указаны относительно центра, но в случае, представленном на рисунке 9, следует применить следующее преобразование:

$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix},$$

где

$$u = \frac{u'}{w'}, v = \frac{v'}{w'}.$$

Алгоритм вычисления координаты точки на изображении:

1. Трехмерная точка преобразуется из глобальной системы координат в локальную путем применения внешней матрицы поворота и переноса;
2. Полученная точка проецируется на плоскость изображения с использованием встроенной матрицы, состоящей из внутренних параметров камеры, таких как фокусное расстояние, координаты оптического центра.

Калибровка камеры с помощью OpenCV

Калибровка – процесс оценки параметров камеры, требуемых для точной взаимосвязи между точкой в пространстве и соответствующей ей точкой на изображении. Для получения более точных сведений о камере для дальнейшего взаимодействия необходимо определить:

- Внутренние параметры камеры, такие как фокусной расстояние, координаты оптического центра, коэффициенты радиального искажения объектива;
- Внешние параметры, связанные с ориентацией (поворотом и переносом) системы координат камеры относительно глобальной системы координат.

$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \boldsymbol{P} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

$$\boldsymbol{P} = \boldsymbol{K} \cdot [\boldsymbol{R}|\boldsymbol{t}]$$

Где **P** – матрица проекции 3x4, состоящая из внутренней матрицы **K** и внешней матрицы [**R**|**t**].

Цель процесса калибровки – нахождение внутренней матрицы, матрицы поворота и вектора переноса для известных трехмерных точек $(X_w, Y_w, Z_w)$ и соответствующие им координаты изображения $(u, v)$. Камера считается полностью откалиброванной, если известны значения внешних и внутренних параметров. Входные данные: изображения с точками, координаты которых известны. Выходные данные: собственная матрица камеры, матрица поворота и переноса каждого изображения. Процесс калибровки производится в соответствии с блок схемой, представленной ниже:

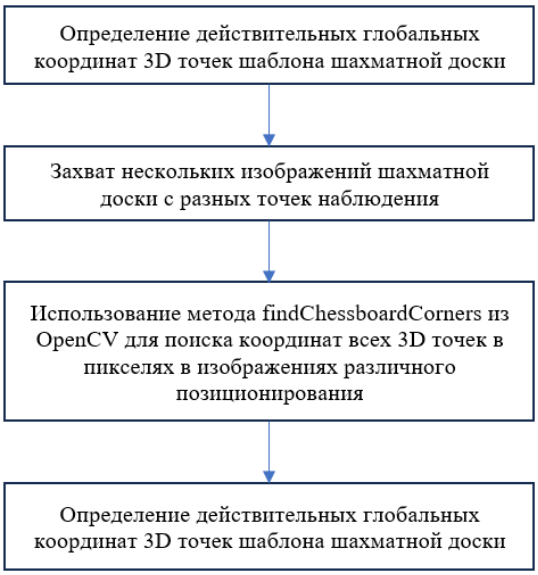


Рисунок 10

Клетки шахматной доски легко различимы на изображении. Угла клеток подходят для локализации, потому что имеют резкие градиенты в двух направлениях, находятся на пересечении линий доски.

OpenCV имеет встроенную функцию findChessboardCorners для поиска координат углов клеток шахматной доски. Данная функция имеет логический тип данных и результатом её работы является значение true или false в зависимости от того, был ли обнаружен шаблон доски. Заключительный этап калибровки состоит в том, чтобы передать 3D точки в глобальных координатах и их 2D координаты в метод calibrateCamera. Пример результата работы программы, реализующей калибровку камеры, представлен на рисунках 11 и 12:

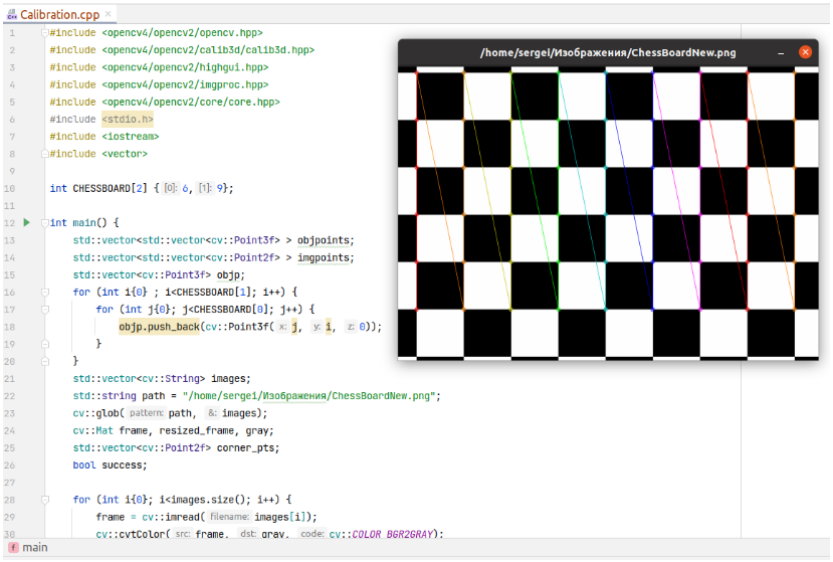


Рисунок 11

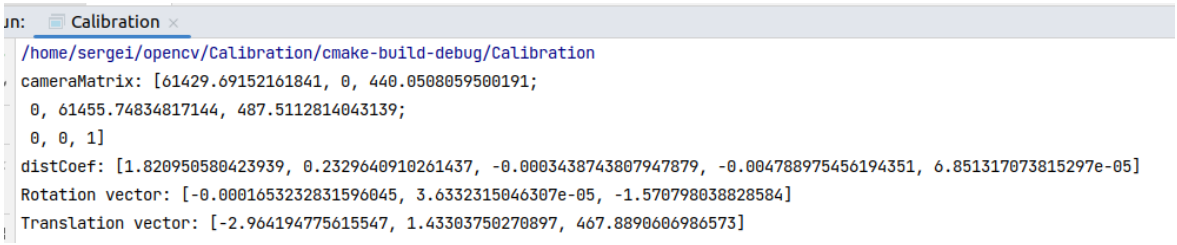


Рисунок 12 – Вычисленные коэффициенты матриц

```
cmake_minimum_required(VERSION 3.25)
project(Calibration)
find_package( OpenCV REQUIRED)
include_directories( ${OpenCV_INCLKUDE_DIRS})
set(CMAKE_CXX_STANDARD 17)
add_executable(Calibration Calibration.cpp)
target_link_libraries( Calibration ${OpenCV_LIBS})
```

Рисунок 13

Полный код программы, реализующий калибровку камеры, представлен в репозитории GitHub в папке Calibration по ссылке из приложения.

Стереовидение и оценка глубины в OpenCV

Для применения данных, полученных с камеры, для ориентации в пространстве и перемещении необходима информация о расстоянии до объектов, находящихся в поле видимости используемой камеры. Для получения данной информации используется карта глубины, построенная по данным с камеры. Карта глубины – это изображение, на котором каждый пиксель содержит информацию о расстоянии от поверхности объекта до точки обзора. Обычные камеры захватывают изображение таким образом, что информация о глубине в явном виде не представлена, поскольку она преобразует 3D-сцену в 2D-изображение. Сопоставление двух изображений, сдвинутых друг относительно друга, называется стереовидением и позволяет получить информацию о расстоянии до объектов на них. Оценка глубины – процесс определения расстояния любого объекта от камеры или расстояния одного объекта от другого в зависимости от того, находятся ли объекты ближе или дальше от камеры или друг друга.

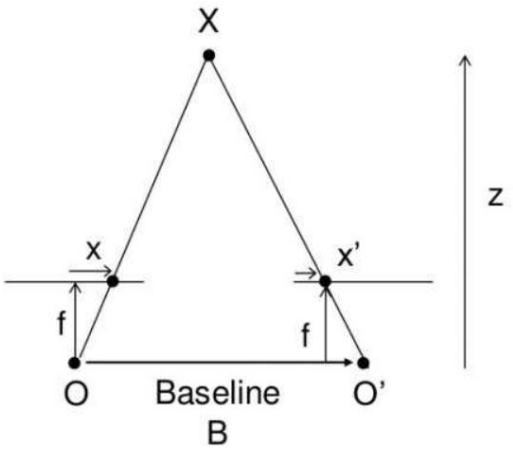


Рисунок 14

Из-за смещения одного изображения относительно другого появляется такое свойство, как несоответствие:

$$disp = x - x' = \frac{Bf}{z}.$$

Глубина точки в сцене обратно пропорциональна расстоянию между соответствующими точками изображения и их центрами камер, откуда можно рассчитать глубину всех пикселей.

В библиотеке OpenCV несоответствие изображения определяется с помощью функции StereoSGBM. Пример работы программы, выполняющей построение карты глубины по двум входным изображениям, представлен на рисунке 15, полный код данной программы представлен в репозитории GitHub в папке Deep_Image по ссылке из приложения.

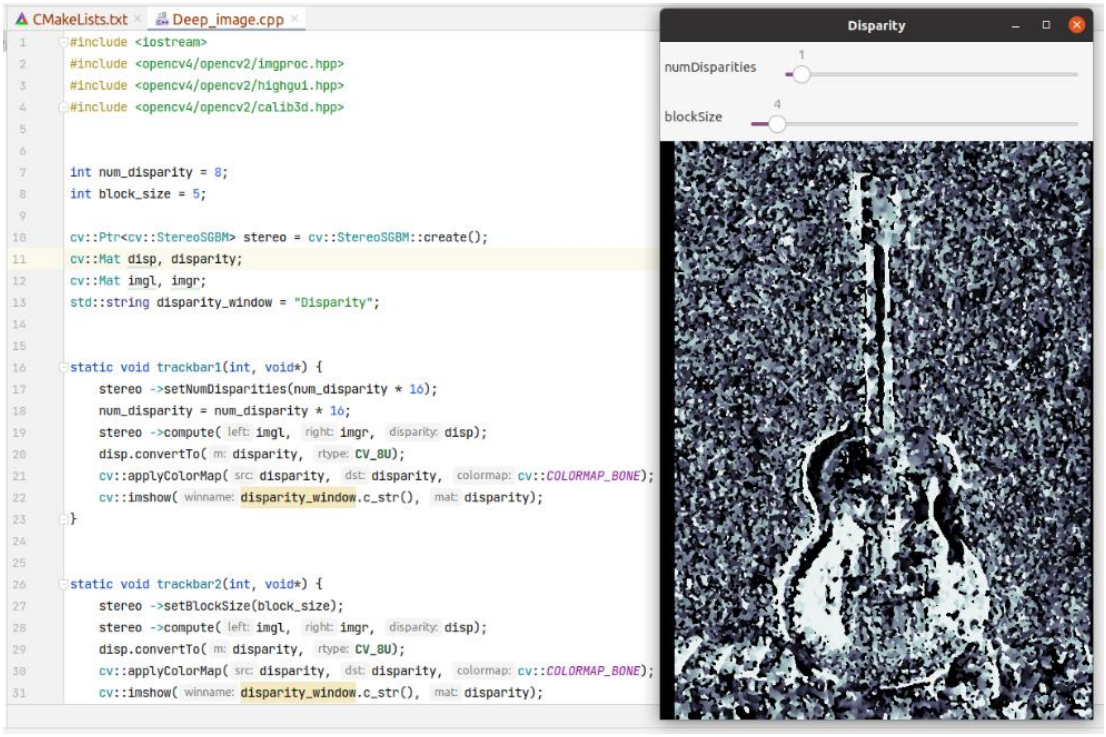


Рисунок 15

```
cmake_minimum_required(VERSION 3.25)
project(Deep_image)
find_package( OpenCV REQUIRED)
include_directories(${OpenCV_INCLUDE_DIRS})
set(CMAKE_CXX_STANDARD 17)
add_executable(Deep_image Deep_image.cpp)
target_link_libraries( Deep_image ${OpenCV_LIBS})
```

Рисунок 16

Использование библиотеки PCL для работы с облаками точек

Для получения трехмерного изображения по двумерным входным данным применяется библиотека PCL (Point Cloud Library). PCL – библиотека алгоритмов для задач обработки облаков точек и обработки 3D-геометрии в трехмерном компьютерном зрении. Пример применения библиотеки OpenCV для вывода облака точек представлен на рисунке 17, файл CMake представлен на рисунке 18. Облако точек выводится в поле pcl_viewer.

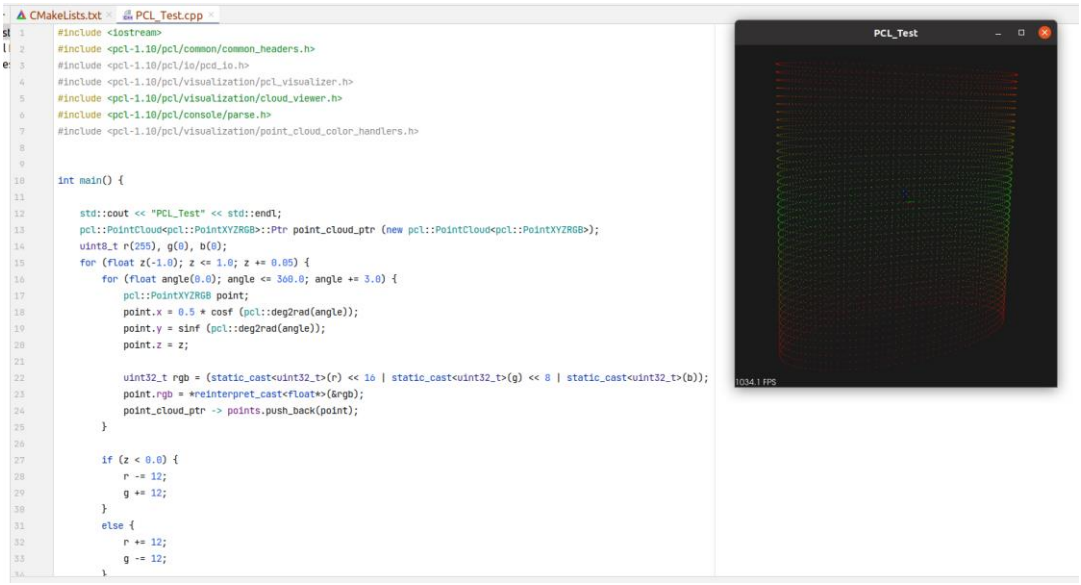


Рисунок 17

```
cmake_minimum_required(VERSION 3.25)
project(PCL_Test)
find_package(PCL 1.9 REQUIRED)
include_directories(${PCL_INCLUDE_DIRS})
link_directories(${PCL_LIBRARY_DIRS})
add_definitions(${PCL_DEFINITIONS})
set(CMAKE_CXX_STANDARD 17)
add_executable(PCL_Test PCL_Test.cpp PCL_Test.cpp)
target_link_libraries(PCL_Test ${PCL_LIBRARIES})
install(TARGETS PCL_Test RUNTIME DESTINATION bin)
```

Рисунок 18

## Заключение

В ходе выполнения задания, выданного для прохождения производственной практики, было проведено изучение основных принципов работы библиотеки OpenCV, предназначенной для применения компьютерного зрения и обработки видео- и фотоматериалов. При обучении для работы с библиотекой были написаны программы, реализующие получение изображения и видеозаписи из файла, получение видеоматериала с встроенной или подключенной камеры, а также рассмотрен метод калибровки камеры с помощью маркера шахматной доски. Подробное описание данных алгоритмов и возможные варианты применения приведены в отчете на страницах 5–14. После получения основных навыков работы с библиотекой была написана программа, реализующая построение карты глубины по двум смещенным изображениям. Полученная карта глубины может быть основой для определения расстояния от камеры до изображенного объекта, поскольку цвет каждого пикселя соотносится с расстоянием от точки до объектива. Далее была установлена библиотека PCL и написана программа для построения и вывода облака точек. Построенное по двум изображениям облако точек может дать полные сведения о местонахождении камеры и расстоянии до объектов.

## Приложение

https://github.com/sergei-pozd25/PozdnyakovSergei/tree/6a3255e9e5bb1c4fd3cb239767f3f6d3a3366bcc/OpenCV_RTC

Список литературы


1. Форсайт, Девид А., Понс, Жан Компьютерное зрение. Современный подход – Издательский дом «Вильямс», 2004. – 928 с.
2. Л. Шапиро, Дж. Стокман Компьютерное зрение – Лаборатория знаний, 2020. – 763 с.
3. R. Hartley, A. Zisserman – Multiple View Geometry in computer vision. – 2nd edition, – Cambridge University Press, 2003. – 673 pg.
4. A. Kaehler, G. Bradski – Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library, – O'Reilly Media, 2016. – 1024 pg.
5. Opencv.org, Camera calibration and 3D reconstruction [Электронный ресурс]. – https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html.
6. Opencv.org, Installation on Linux [Электронный ресурс]. – https://docs.opencv.org/4.x/d7/d9f/tutorial_linux_install.html.
7. GitHub.com OpenCV [Электронный ресурс]. – https://github.com/opencv/opencv/tree/4.7.0.
8. LearnOpenCV.com Geometry of Image Formation [Электронный ресурс].– https://learnopencv.com/geometry-of-image-formation/.
9. LearnOpenCV.com Camera Calibration Using OpenCV [Электронный ресурс]. – https://learnopencv.com/camera-calibration-using-opencv/.
10. LearbOpenCV.com Stereo Camera Dept Estimation with OpenCV (Python, C++) [Электронный ресурс]. – https://learnopencv.com/depth-perception-using-stereo-camera-python-c/.