|  |
| --- |
| **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 СПЕЦИАЛЬНОЕ МАШИНОСТРОЕНИЕ

Кафедра РОБОТОТЕХНИЧЕСКИЕ СИСТЕМЫ И МЕХАТРОНИКА

**Отчёт К ПРОЕКТУ**

НА ТЕМУ:

***«Решение задач управления для 3-х степенного манипулятора»***

Студент СМ7И-33М \_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_Разананирина\_Ж.К.

(Группа) (подпись, дата) (Фамилия И.О.)

Ркуоводитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_Шереужев М.А\_\_

(подпись, дата) (Фамилия И.О.)

**Москва**

**2021 г.**

**Оглавление**

[1. Функциональная схема системы распознавания яблок 3](#_Toc84258147)

[2. Разработка нейронных сетей для распознавания яблок 4](#_Toc84258148)

[3. 3D зрение на основе стереопары 6](#_Toc84258149)

[3.1. Модель камеры 7](#_Toc84258150)

[3.2. Фокусное расстояние 8](#_Toc84258151)

[3.3. Искажение объектива 9](#_Toc84258152)

[3.4. Калибровка с OpenCV 10](#_Toc84258153)

[3.5. Стерео изображение 12](#_Toc84258154)

[3.6. Триангуляция 13](#_Toc84258155)

[3.7. Эпиполярная геометрия 14](#_Toc84258156)

[4. Определение расстояния от камер 15](#_Toc84258157)

[4.1. Создание карты диспаратности 15](#_Toc84258158)

[4.2. Построение облака точек 17](#_Toc84258159)

# Функциональная схема системы распознавания яблок

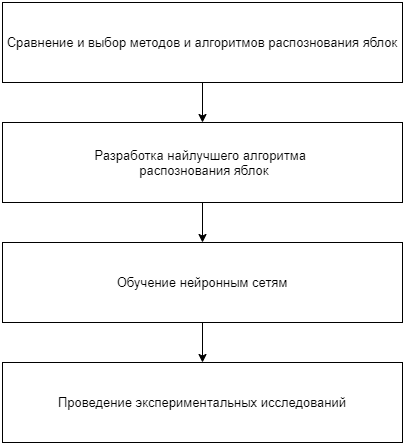
На рисунке 1 представлена функциональная схема системы.



*Рис 1. Функциональная схема системы захвата яблок*

Левая камера сканирует его окружения. При обнаружении яблок, система определяет его область, затем проверяет что область находиться в правой части изображения левой камеры (при построении карты диспаратности левая часть теряется). Затем правая камера снимает то же сцену. Поскольку созданию карты диспаратности требует черно-белые изображения, обрабатываем их. Для визуализации положения яблока в пространстве, создаем облаку точек. После этого определяем расстояние между яблоком и камерами. Зная положения яблока, манипулятор может захватить его.

# Разработка нейронных сетей для распознавания яблок



*Рис 2. Разработка нейронных сетей для распознавания яблок*

В рамках данной работы мы будем сравнивать разные алгоритмы распознавания объектов и разрабатывать наилучший алгоритм для распознавания яблок.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Название сети** | **Входное разрешение** | **Метод обучения** | **Выход сети** | **Скрорсть**  **(FPS)** | **Точность распознавания** |
| R-CNN | 227 x 227 | SGD,BP | {c,x,y,h,w} | 0.5 | 79 |
| YOLO | 640 x 520 | SGD | {c,x,y,h,w} | 30 | 88 |
| SDD | 300 x 300 | SGD | {c,x,y,h,w} | 46 | 83 |

Где: FPS(Frame Per Second) – число кадров в секунду

SGD (Stochastic gradient descent) - Стохастический градиентный спуск

BP – (Back Error Propagation) – алгоритм обратного распространения ошибки

c – класс объектов

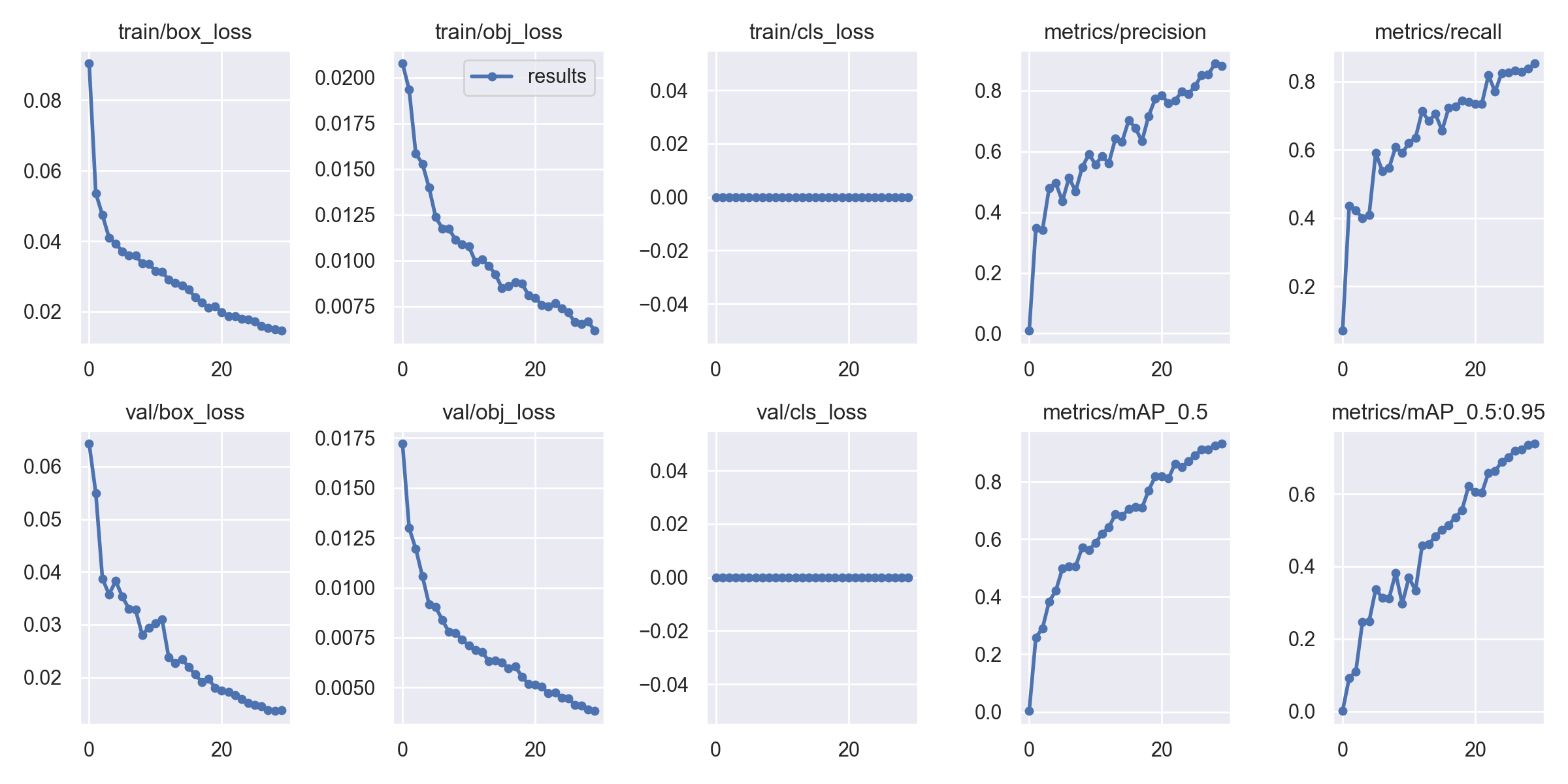
x,y – центр рекомендуемой области

h,w – длина и ширина рекомендуемой области

Из таблицы очевидно, что YOLO обеспечивает более точнее распознавание среди этих нейронных сетей. Более того он позволяет снимать достаточное количество кадров в секунду с хорошим входным разрешением.

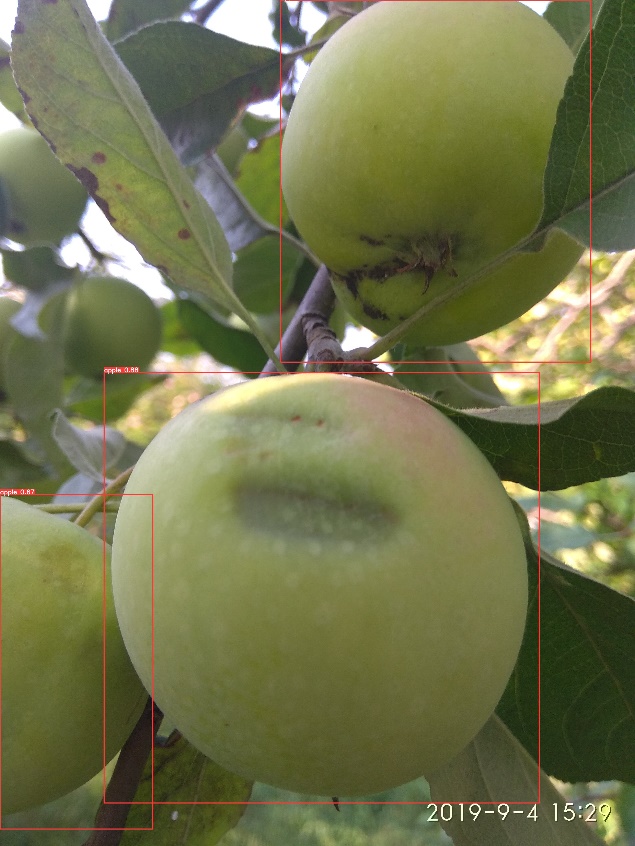
Нейронных сетей обучали с “MS COCO dataset”. MS COCO dataset — это набор данных для крупномасштабного обнаружения объектов, сегментации и субтитров, опубликованный корпорацией Microsoft. Он содержит миллионов изображений. Из этого набора данных 1171 изображении яблок были получены.

Обучение нейронных сетей осуществлялось в течение 30 эпох. Результаты обучения представлены на рисунке 3.



*Рис 3. Результаты после обучения нейронных сетей на распознавание яблок*

Пример результатов распознавания яблок в реальных условиях представлен на рисунках ниже.

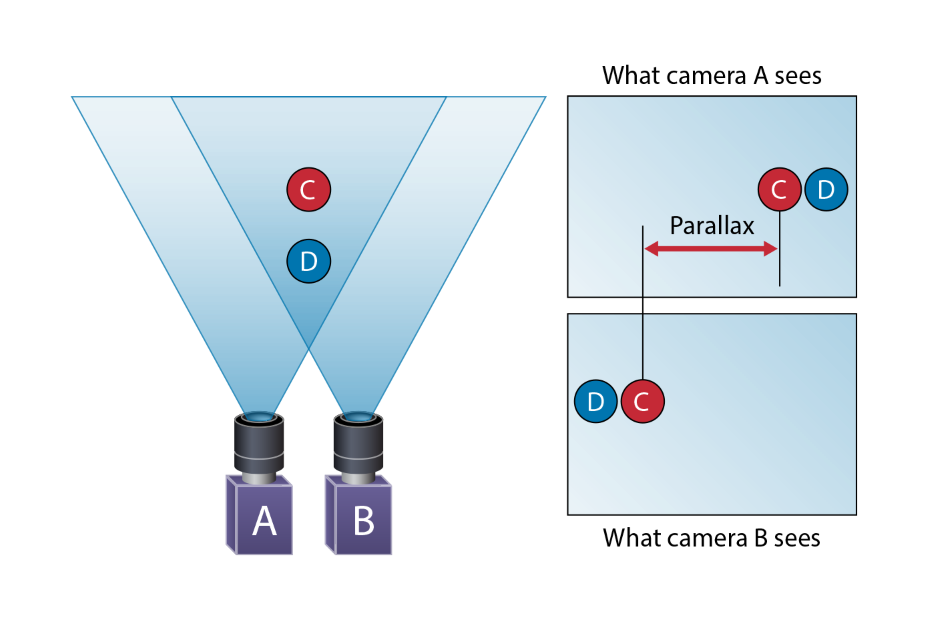
 

*Рис 4. Пример результатов распознавания яблок в реальных условиях*

# 3D зрение на основе стереопары

Для распознавания объектов манипулирования решено использовать систему стереозрения.

**Стереозрение** - это процесс, в ходе которого две камеры, смещенные горизонтально друг от друга, используются для получения двух разных видов сцены, аналогично человеческому бинокулярному зрению. Сравнивая эти два изображения, можно получить информацию об относительной глубине в виде карты диспаратности, которая кодирует разницу в горизонтальных координатах соответствующих точек изображения. Значения на этой карте диспаратности обратно пропорциональны глубине сцены в соответствующем местоположении пикселя.



*Рис 5. Стереозрение*

Чтобы идентифицировать цель, нам нужна одна камера, а для измерения расстояния до нее нам нужны как минимум две камеры (стереозрения), поэтому мы решили использовать стереокамеру для получения координаты цели.

## Модель камеры

Использование камер с линзами вызвает два разных типа искажения:

• радиальная деформация

• тангенциальное искажение

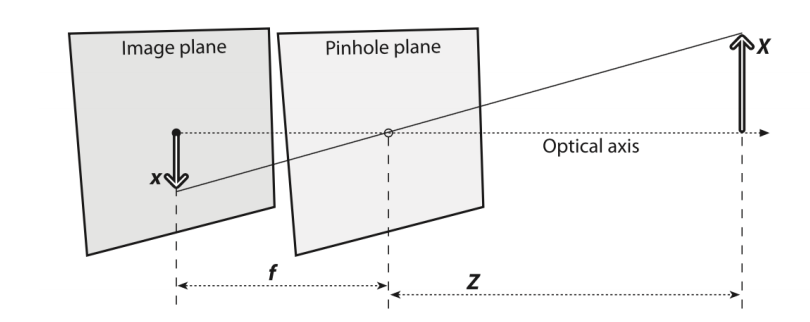
Радиальное искажение обусловлено формой самого объектива, а тангенциальное искажение - геометрией камеры. Затем изображения можно исправить с помощью математических методов.

Процесс калибровки позволяет сформировать модель геометрии камеры и модель объектива. Эти модели формируют внутренние параметры камеры.

## Фокусное расстояние

Относительный размер изображения, проецируемого на поверхность в камере, зависит от фокусного расстояния. В модели точечного отверстия фокусное расстояние - это расстояние между отверстием и областью, на которую проецируется изображение.

Тогда теорема Фалеса дает:



*Рис 6. Проектируемый объект*

При:

• : изображение объекта (знак минус происходит из-за того, что изображение перевернуто)

• : размер объекта.

• : расстояние от отверстия до объекта.

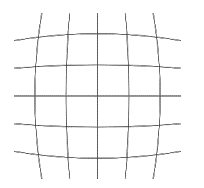
• : фокусное расстояние, расстояние от отверстия до изображения

Поскольку линза не идеально отцентрирована, вводятся два параметра, Cx и Cy для горизонтального и вертикального смещения линзы соответственно. Фокусные расстояния по осям X и Y также различаются, поскольку область изображения прямоугольная. Это дает следующую формулу для положения объекта на поверхности.

Таким образом, проецируемые точки реального мира на поверхность изображения можно смоделировать следующим образом. M - Внутренняя матрица.

## Искажение объектива

Как обсуждалось ранее, есть два типа искажения. Радиальное искажение, возникающее из-за формы объектива, и тангенциальное искажение, вызванное процессом сборки камеры.В оптическом центре нет радиального искажения, и оно становится все больше и больше по мере приближения к краям.

**

*Рис 7. Искажение объектива*

На практике это искажение остается небольшим; достаточно сделать разложение Тейлора до третьего члена. В результате получается следующая формула.

)

)

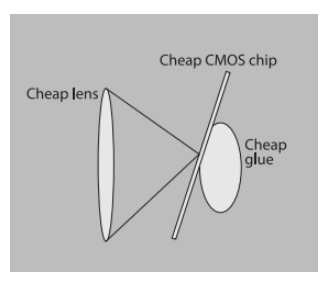
Где:

x и y - координаты исходной точки на поверхности изображения, и положение исправленной точки вычисляется с их помощью.

Также существует тангенциальное искажение, потому что линза построена не идеально параллельно поверхности изображения. Чтобы исправить это, вводятся два дополнительных параметра, p1 и p2.

)]

)]



*Рис 8.. Тангенциальное искажение*

## Калибровка с OpenCV

Библиотека OpenCV позволяет нам рассчитывать внутренние параметры с помощью определенных функций, этот процесс называется калибровкой. Это стало возможным благодаря различным видам шахматной доски.



*Рис 9. Фотография во время съемки изображений для калибровки*

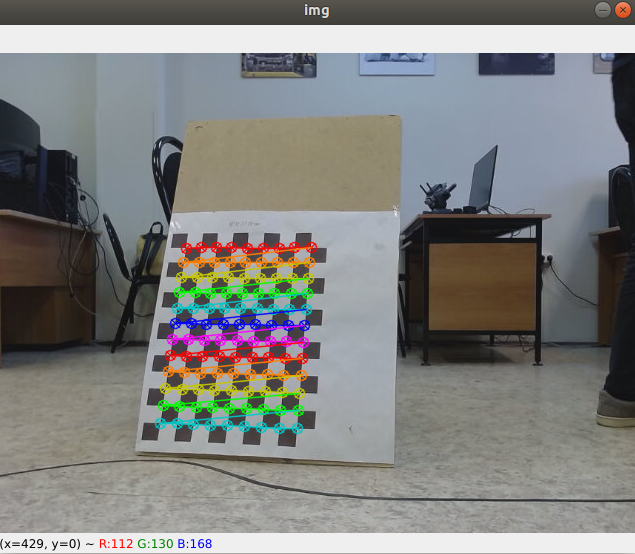
Программа для фотосъемки для последующей калибровки называется get\_images.py

Эти изображения используются позже для калибровки в основной программе single\_camera\_calibration.py.

OpenCV рекомендует иметь не менее 10 изображений для каждой камеры, чтобы получить хорошую калибровку. Получились хорошие результаты - по 15 снимков на каждую камеру.

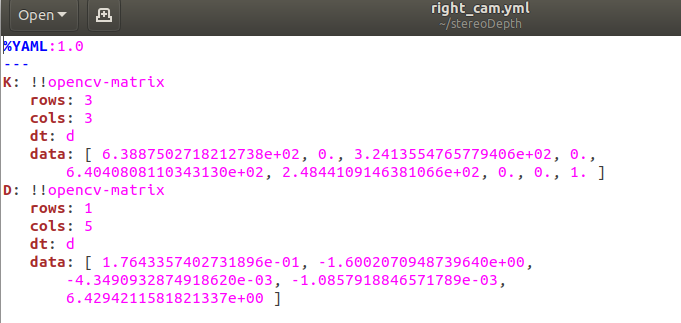
Чтобы откалибровать камеры, код Python ищет углы шахматной доски на каждом изображении для каждой камеры с помощью функции OpenCV: cv2.findChesssboardCorners

Положение углов для каждого изображения затем сохраняется в одном векторе изображения, а точки объекта для 3D-сцены сохраняются в другом векторе. Эти Imgpoints и Objpoints затем используются в функции cv2.calibeCamera (), которая возвращает на выходе матрицу камеры, коэффициенты искажения, векторы поворота и сдвига.



*Рисунок 10. Процесс калибровки*

После калибровки с помощью OpenCV получаем для наших камер матрицы:



*Рисунок 11. Матрица М для правой камеры.*

## Стерео изображение

Стереозрение позволяет распознавать глубину изображения, проводить измерения на изображении и выполнять трехмерную локализацию. Для этого, необходимо найти точки, которые совпадают между двумя камерами. Отсюда вы можете определить расстояние между камерой и точкой. Геометрия системы используется для упрощения расчета.

Создание стереоизображения состоит из четырех этапов:

1. Устранение радиальных и тангенциальных искажений математическими расчетами. Это дает вам неискаженные изображения.

2. Исправление угол и интервал между изображениями. На этом этапе оба изображения могут быть копланарными по оси Y, что упрощает поиск соответствий, и нужно искать только по одной оси (а именно по оси X).

3. Найти одну и ту же функцию на правом и левом изображениях. Это дает карту несоответствия, которая показывает различия между изображениями по оси X.

4. Последний шаг - триангуляция. Карта диспаратности преобразуется в расстояния путем триангуляции.

То есть:

Шаг 1: устранение искажения

Шаг 2: Исправление

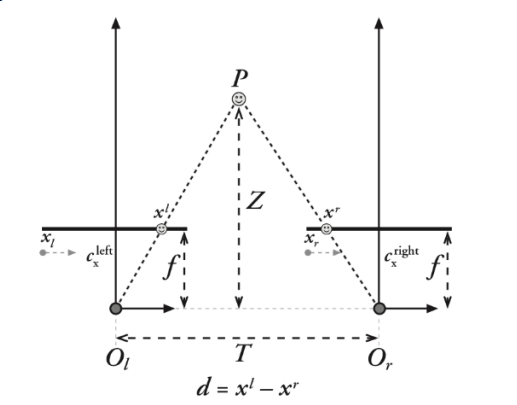
Шаг 3. Обнаружение одной и той же функции на обоих изображениях

Шаг 4: триангуляция

## Триангуляция

На последнем этапе, триангуляции, предполагается, что оба проекционных изображения компланарны и что горизонтальная строка пикселей левого изображения выровнена с соответствующей горизонтальной строкой пикселей левого изображения.

Используя предыдущие гипотезы, теперь можно построить следующую картину.



*Рисунок 12. Триангуляция.*

Точка находится в окружающей среде и отображается на и на левом и правом изображениях с соответствующими координатами и . Это позволяет нам ввести новую величину - дисп аратность: . Видно, что чем дальше точка , тем меньше размер Таким образом, несоответствие обратно пропорционально расстоянию.

Для расчета расстояния можно использовать следующую формулу: существует нелинейная зависимость между несоответствием и расстоянием. Когда несоответствие близко к 0, небольшие различия диспаратности приводят к большим различиям расстояний. Это наоборот, если несоответствие велико. Небольшие различия в несоответствии не приводят к слишком большим различиям в расстояниях. Из этого можно сделать вывод, что стереозрение имеет высокое разрешение по глубине только для объектов, находящихся близко к камере.

Этот метод работает только в том случае, если конфигурация стереокамеры идеальна. Однако на самом деле это не так. Следовательно, левое и правое изображения математически выравниваются параллельно. Конечно, камеры должны быть физически расположены параллельно, хотя бы приблизительно.

## Эпиполярная геометрия

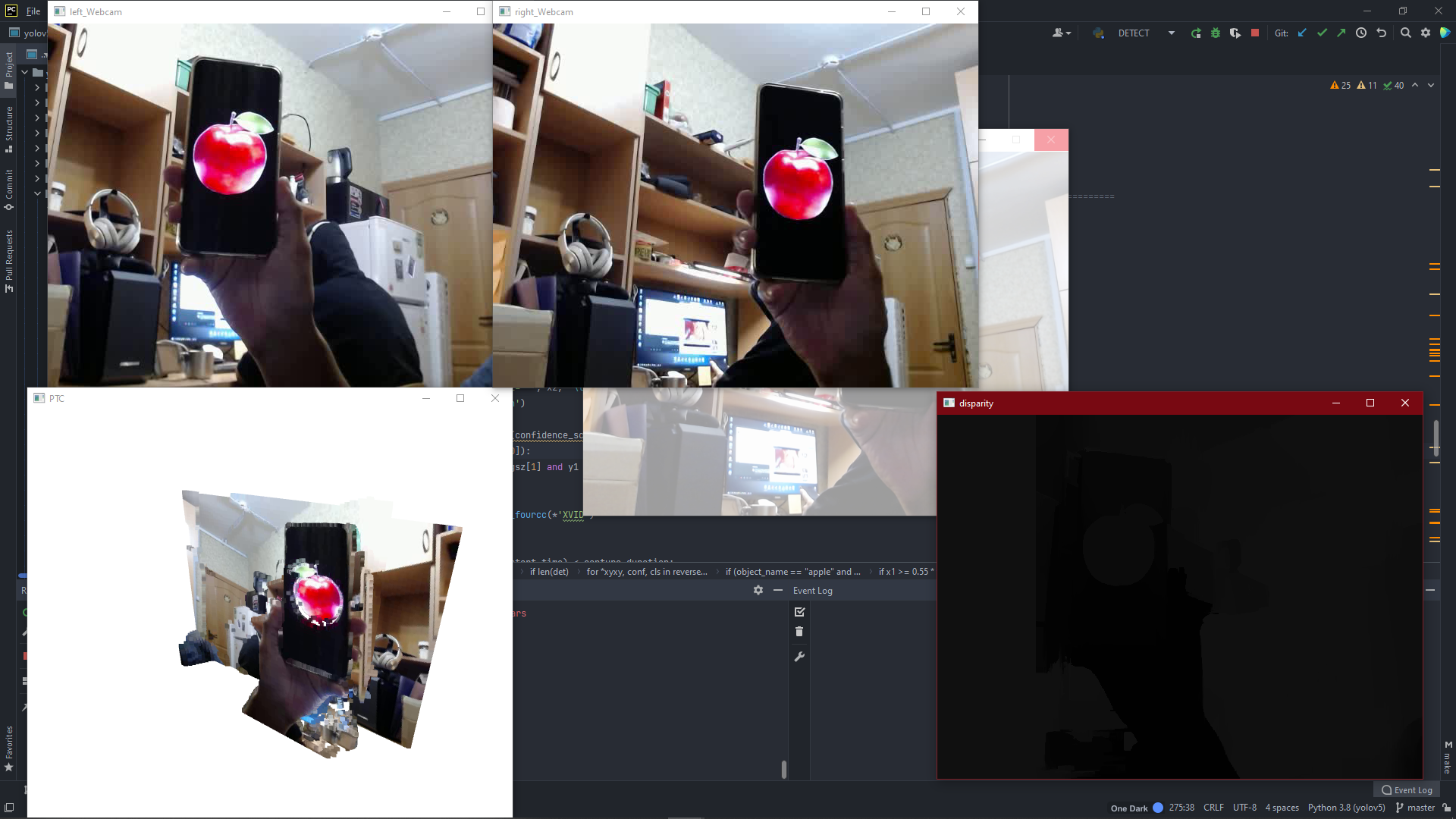
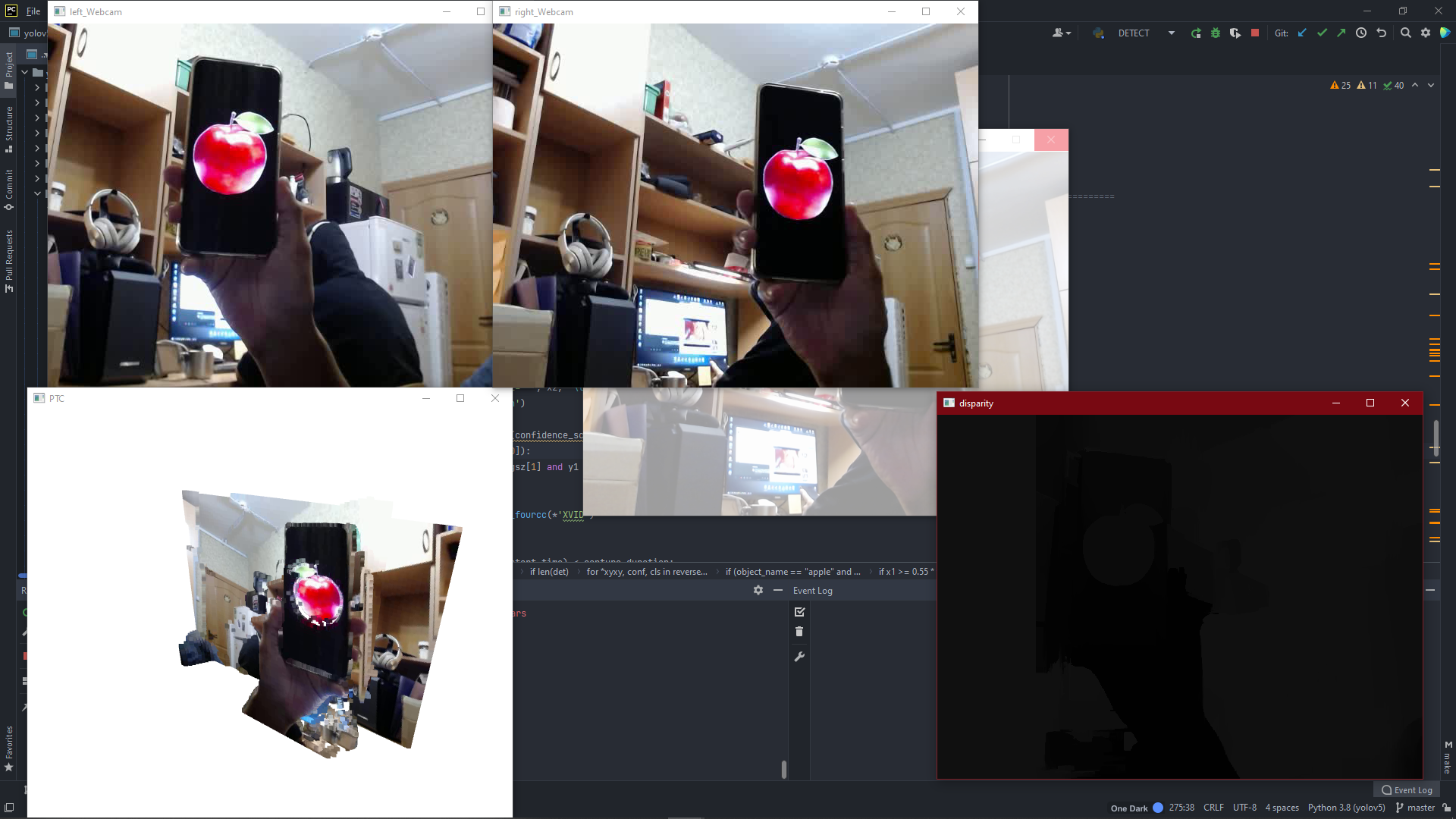
На картинке выше показана модель несовершенной стереокамеры, состоящая из двух моделей камер-обскур. Пересечение линии центров планирования проекта (Ol, Or) с проекционными плоскостями создает эпиполярные точки el и er. Линии (pl, el) и (pr, er) называются эпиполярными линиями. Изображение всех возможных точек точки на плоскости планирования - это эпиполярная линия, которая лежит на другой плоскости изображения и проходит через эпиполярную точку и искомую точку. Это позволяет ограничить поиск точки одним измерением, а не всей плоскостью. Итак, можно резюмировать следующие моменты:

* Каждая трехмерная точка в поле зрения камеры включена в эпиполярный план.
* Элемент одного уровня должен находиться на соответствующей эпиполярной линии другого уровня (эпиполярное состояние).
* Двумерный поиск соответствующего объекта преобразуется в одномерный поиск, если эпиполярная геометрия известна.
* Порядок точек сохраняется, т.е. две точки A и B находятся в том же порядке на эпиполярной линии одного уровня, что и на другом уровне.

# Определение расстояния от камер

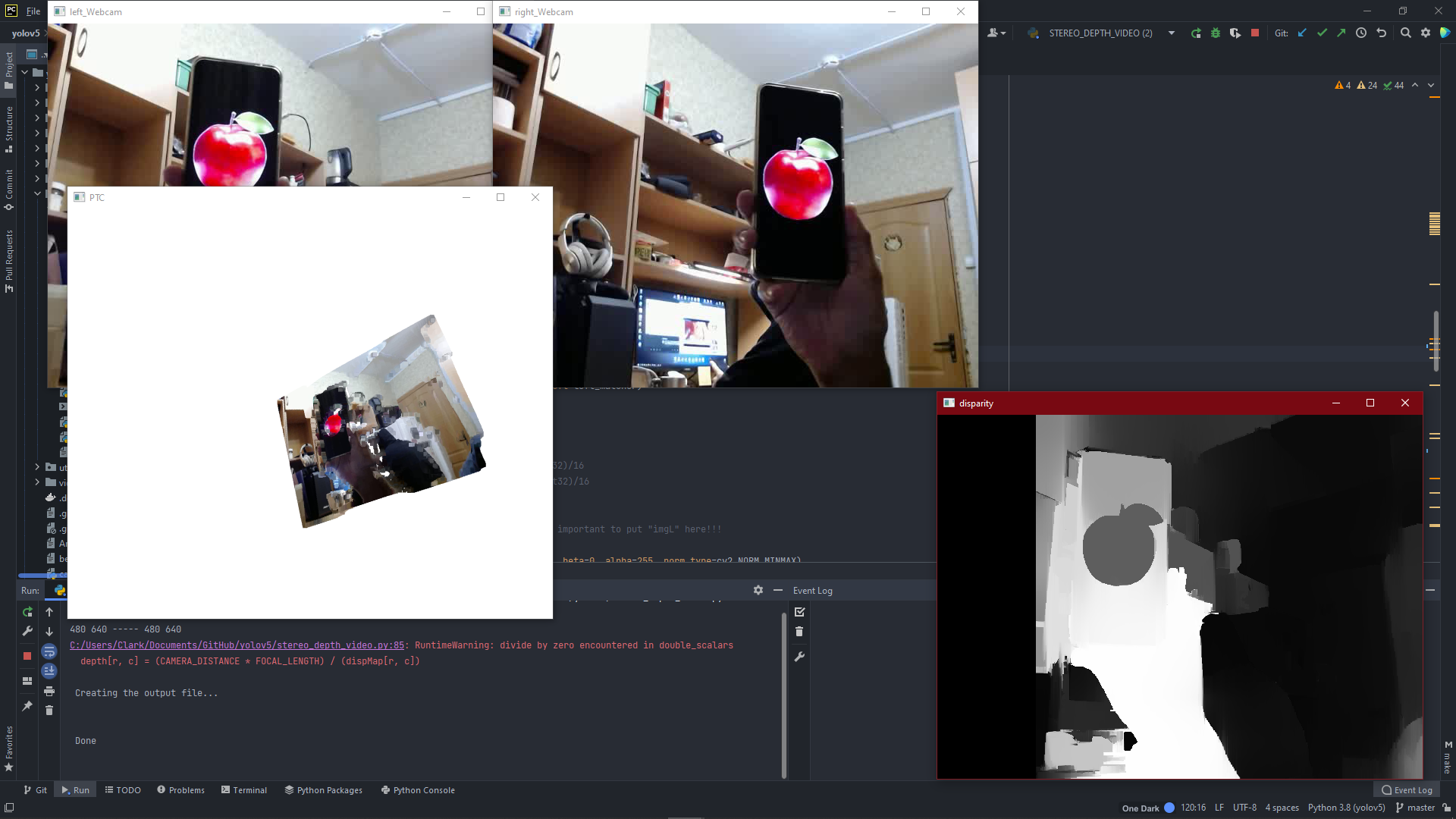
## 4.1. Создание карты диспаратности

Библиотека openCV позволяет нам использовать SGBM и WLS для построения карты диспаратности из двух изображения (stereo) с помощью двух функций: cv2.StereoSGBM\_create() и cv2.ximgproc.createDisparityWLSFilter().

*Рисунок Два RGB изображения той же сцены.*

Карта диспарности этой сцены была построена на рисунке.



*Рисунок Карта диспаратности*

Как видно выше, вычисленная карта диспаратности представлена в оттенках серого, и мы можем видеть, что диспаратности объектов, находящихся ближе к камере, больше (белее) по сравнению с тем, что в заде.

Следовательно, результаты значений диспаратности совершенно не соответствуют фактическим глубинам.

Маскирование этих значений (путем установки высоких значений диспропорций) дает нам более четкую карту диспропорций и глубины.

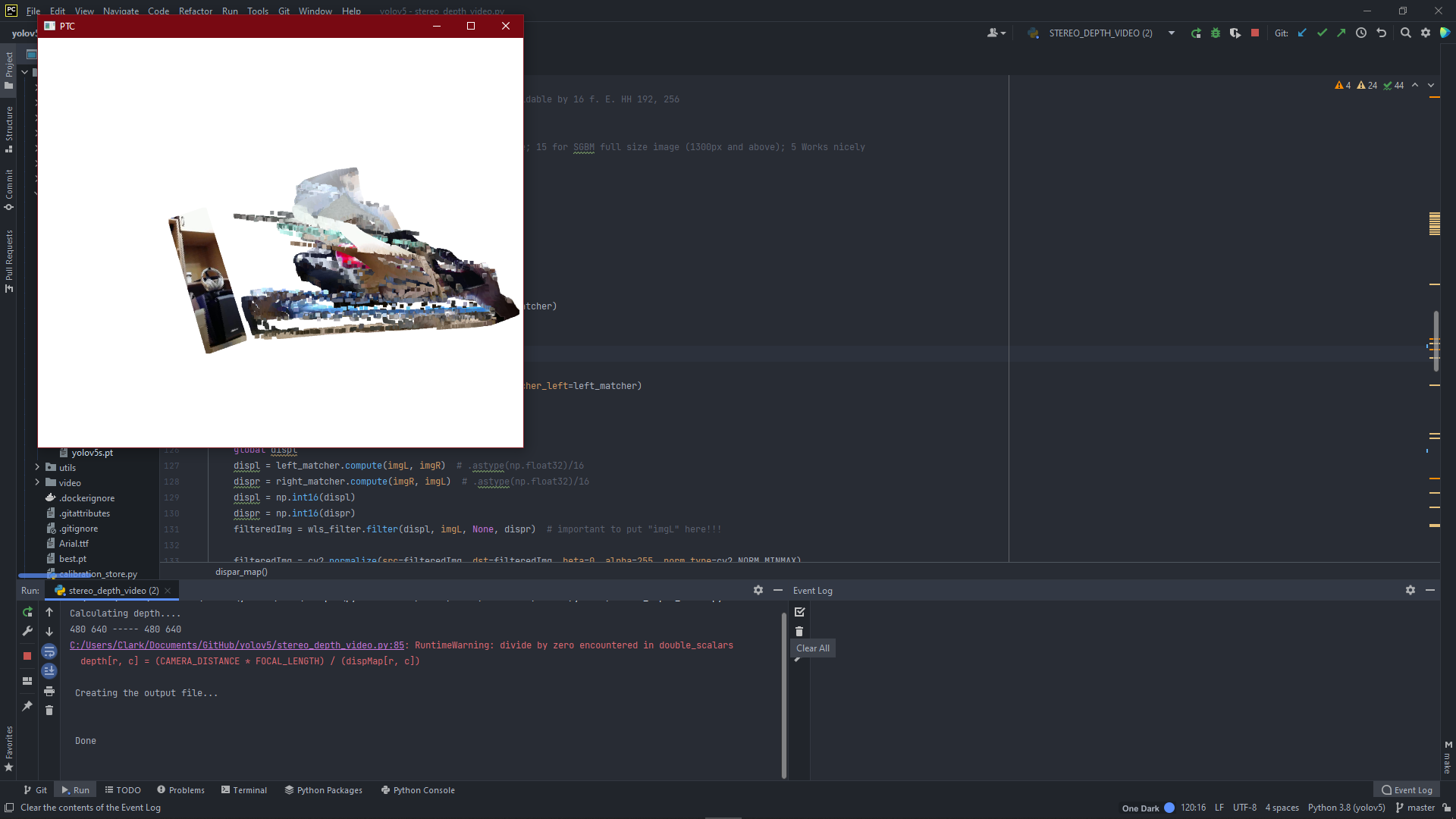
Затем мы вычисляем точные координаты каждого пикселя в видимой сцене, учитывая смещения осевых линий x и y. Эти координаты позже используются для построения облаков точек используя следующие формулы:

*; ;* 

## 4.2. Построение облака точек

Учитывая координаты, вычисленные выше, у нас есть расположение каждого пикселя в трехмерном пространстве для облака точек.

Теперь необходимо определить точные значения R G B для каждого пикселя, который нужно залить. Значения можно получить, оценив исходное цветное изображение в уменьшенном масштабе. Следует учитывать, что OpenCV представляет изображения как BGR, поэтому необходимо инвертировать индексы при оценке конкретных значений RGB. СОздается файл слоя (формат многоугольника), в котором каждая строка содержит соответствующие значения X, Y, Z (координаты) и значения R, G, B. Выходной файл ply затем может быть нанесен на график с помощью любых плоттеров файлов ply , для этого используется библиотека open 3D.



Итак можем вычислить расстояние яблока от камер.

