Для работы кода на Python используются библиотеки:

* *(*
* ()
* ()

Уже проведена калибровка камеры:

* - фокусные расстояния в пикселях
* - координаты главной точки (оптического центра)
* - коэффициенты для коррекции дисторсии (k1, k2, p1, p2, k3)

**Коррекция дисторсии точек**

Используется функция () для исправления искажения объектива для отдельных точек (для углов первой доски, для углов второй доски, для центра объекта)

**Обнаружение шахматных досок**

Исходное цветное изображение преобразуется в оттенки серого.

С помощью функции () находит углы первой шахматной доски по параметрам поиска углов ().

Функция () уточняет положение углов с субпиксельной точностью:

* + Размер окна для уточнения: 11x11 пикселей
  + Использует алгоритм наименьших квадратов

Дальше происходит поиск второй шахматной доски (с маскированием первой)

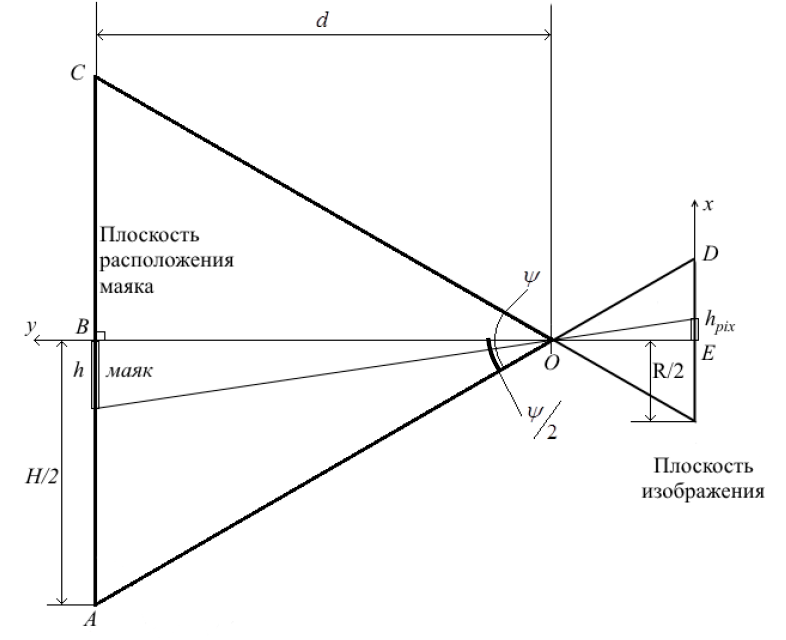
* Создается маска, закрашивающая область первой доски черным цветом (0)
* Применяется операция AND для исключения этой области из поиска (**)**)

Поиск углов второй доски происходит по такому же алгоритму, как и поиск углов первой доски.

**Расчет расстояния до доски**

Расчет расстояния до доски происходит в функции ()

*О* – расположение объектива телекамеры, плоскость *АВ* –плоскость, в которой расположен маяк, *ED* – плоскость формирования изображения или матрица телекамеры.



Формула основана на соотношении:

R– разрешение (число пикселей) изображения по высоте, пикс.;

– относительна высота маяка на изображении, пикс.;

*H* – расстояние в плоскости маяка между краями изображения, попадающего в кадр, cм;

*h* – реальная высота маяка, cм.

Расстояние *H* может быть выражено через тангенс половины угла обзора объектива и расстояние от камеры до плоскости маяка:

– угол обзора объектива телекамеры, градус,

d – расстояние камера- плоскость маяка, cм.

Формула расстояния:

Другой способ:

Рассчитывает евклидово расстояние в 3D пространстве по формуле

Координаты получаются из (вектор перевода/position vector) полученного после выполнения функции **cv2.solvePnP().**

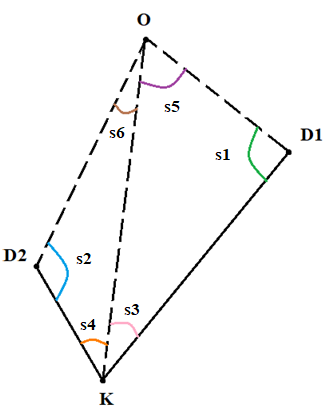
**Обнаружение кружки с YOLOv8**

* Используется модель YOLOv8 для обнаружения объектов
* Фильтруются результаты, оставляя только кружки (класс 41 в COCO)
* Находится центр обнаруженной кружки и корректируется дисторсия

**Расчет углов и расстояния до объекта**

Если найдены обе доски и кружка:

* Вычисляются углы между центрами досок, кружкой и оптическим центром камеры



Координаты

*-Объекта*

*– Доски 2*

*– Доски 1*

*- Камеры*

**Расчет углов для доски 1.**

***Угол s3***

***Угол s5***

***Угол s1***

***Расчет углов доски 2.***

***Угол s4***

***Угол s6***

***Угол s2***

* Используется теорема синусов для определения расстояний:

D1 - расстояние до кружки на основе первой доски

D2 - расстояние до кружки на основе второй доски

**Другой способ использовать теорему косинусов:**

**Третий способ:**

Через пропорцию.