Алгоритм перевода точек из 2D в 3D

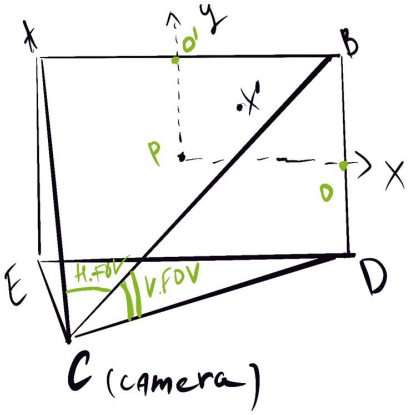
Входные данные:
- Массив контрольных точек kp
- Соответствующий массив дистанции к каждой контрольной точке
- Размер изображения (ширина, высота в пикселях)
- *Дополнительно: само изображение. Передается для того, чтобы брать цвет для каждой контрольной точки с изображения.*

Выходные данные:
- Массив преобразованных точек. В каждой ячейке хранится массив координат точки.
- *Дополнительно: массив с цветами точек.*

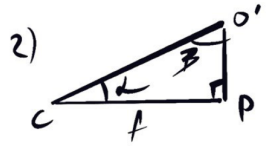
Константы:
$f = 10.0095$ м - фокусное расстояние нашей камеры.
$H.FOV = 90^\circ$
$V.FOV = 59^\circ$

Цикл для каждой контрольной точки изображения:
Кратко: вычислим векторCX' от камеры до точки на изображении и найдем коэффициент его растяжения, такой, что значение координаты z равнялось вычисленной дистанции до данной точки.

Подробно:
Далее последуют кривые рисуночки от Арины.



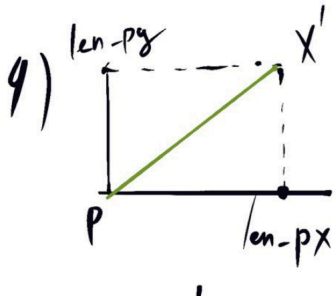
На рисунке изображена камера C
$ABDE$ — изображение, которое захватывает камера.
1) Р/м треугольник CPO
$\beta = \frac{H.FOV}{2} = 45^\circ \rightarrow \alpha = 45^\circ$
$\rightarrow PO = PC = f$



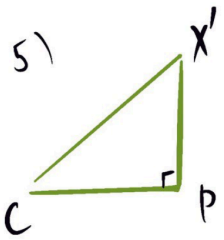
2) Р/м CPO'. $\alpha = \frac{V.FOV}{2} = 29^\circ 5'; \quad \beta = 90 - \alpha$
$PO' = f\frac{\sin\alpha}{\sin\beta}$ по т.синусов.

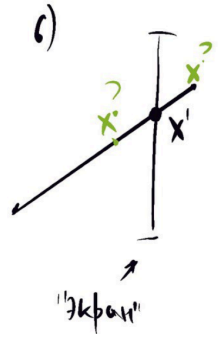
3) Координаты точки x' в данный момент есть только в пикселях.
Известны PO и PO' и размеры изображения im_x, im_y
$len\_px = \frac{x'[0]}{im\_x/2}$ (в метрах)

$$len\_py = \frac{x'[1]}{im\_y/2}$$(в метрах)



4) $PX' = \sqrt{len\_py^2 + len\_px^2}$ (м)





5) р/м CPX'

$CX' = \sqrt{PX'^2 + f^2}$ - вектор от камеры до точки на изображении. Однако физически точка X может располагаться ближе или дальше от "экрана" изображения, который находится на фокусном расстоянии.

6) Пусть $DIST$ - вычисленная ранее дистанция до данной точки.

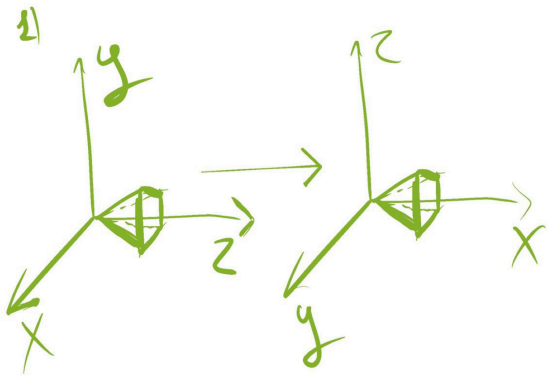
Тогда $k = \frac{DIST}{CX'}$ - коэффициент растяжения.

Искомые координаты точки: $CX = k * N$, где $N = \begin{pmatrix} len\_px \\ len\_py \\ CX' \end{pmatrix}$

Алгоритм перевода точек из локальных координат в глобальные:

Входные данные:
- Массив 3д точек в локальных координатах дрона,
- Вращение дрона на данном кадре (Pitch, yaw, Roll)
- Смещение дрона

Выходные данные:
- Массив точек в глобальных координатах



В цикле для кадой точки:

Переводим координаты с помощью матрицы $\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$

В коде просто делаем следующее:

```
tmp = np.array([[point[1]],

       [point[2]],

       [point[0]]])
```

Находим матрицу поворота

```
def createRotationMatrix(pitch, yaw, roll):
    Ryaw = np.array([[np.cos(yaw), -np.sin(yaw), 0],
                     [np.sin(yaw), np.cos(yaw), 0],
                     [0,    0,   1]])
    Rpitch = np.array([[np.cos(pitch), 0, np.sin(pitch)],
                       [0,  1,  0],
                       [-np.sin(pitch), 0, np.cos(pitch)]])
    Rroll = np.array([[1,    0,    0],
                     [0, np.cos(roll), -np.sin(roll)],
                     [0, np.sin(roll), np.cos(roll)]])
    R = np.dot(np.dot(Ryaw, Rpitch), Rroll)
    return R
```

Умножаем матрицу поворота на координаты

```
tmp = np.dot(R, tmp)
```

Добавляем смещение к координатам

```
tmp = tmp+translation.reshape(3,1)/100
```

Прим.: надеюсь к этому моменту вы не пробили себе лицо фейспалмами. (Простите, надеюсь все хорошо…)