Итоговое задание

Задача: для последовательности кадров с RGB-D сенсора найти трехмерные позиции трех ключевых точек (головы, левого и правого запястья) на каждом кадре последовательности.

Открытые тестовые последовательности выложены в папку data/final. Там же лежат zip архивы с теми же данными для удобного скачивания.

Всего имеется 5 кейсов - папки test1-test5.
Каждая папка содержит:
-Depth – последовательность карт глубины;
-RGB – последовательность соответствуюших RGB изображений;
-Mask – последовательность соответствующих бинарных масок человека;
-data.csv – файл, содержащий референсные трехмерные позиции суставов (в первой строке – названия столбцов: FrameID и названия суставов; в каждой следующей – id кадра и позиции).

Описание кейсов

1) test1.
Звездочка.
- нет самоперекрытий
- простой кейс: левая рука справа от тела, прававя - слева

2) test2
Рука согнута в локте и поднята над головой + движение
-все еще без самоперекрытий, но руки находятся не по бокам туловища

3) test3
Рука согнута в локте и выставлена перед туловищем + отрезание рукой верхней части
- самоперекрытия сегмента
- без самоперекрытий рук

4) test4
Скрещенные руки перед корпусом, смена дальней/ближней руки + вращение корпуса
-самоперекрытие рук

5) test5
-перекрытия
-разворот и уход вдаль

Нужно находить 3d координаты точек скелета: head, wrist.L, wrist.R.

Параметры камеры и преобразование координат

$u = c_x + X/Z * f_x$,
$v = c_y - Y/Z * f_y$,
где $c_x = 320$, $c_y = 240$, $f_x = f_y = 575.7$, (u, v) - проективные координаты, (X, Y, Z) – 3d-координаты.

Решение будет проверяться на закрытом датасете (по несколько тестов на каждый из пяти кейсов).

Метрика качества

Для каждого теста будет считаться средний процент попадания найденной позиции в окрестность радиуса *100 мм* от истинной позиции для каждой из трех ключевых точек.
В качестве итогового результата будет выступать средний процент попадания по всем ключевым точкам по всем тестовым последовательностям.

Требования

Программа должна быть реализована на языке **python**.
Ограничение по времени (на кадр): **1000 мс**;
На первом кадре последовательности допускается превышение времени работы в 4 раза.

Основной скрипт должен называться **solver.py**. Он должен лежать рядом с папкой data (в папке /opt/notebooks/).
В нем должен быть определен класс **Solver**.

Для создания объекта класса в проверяющем скрипте будет использоваться **конструктор без параметров**:

```
solv = Solver()
```

В классе должен быть определен метод для обработки очередного кадра с именем **process**.

```
def process(self, data)
```

В кортеже data будут поступать новые данные: depth, mask, rgb – уже считанные изображения.

```
depth, mask, rgb = data
```

Метод должен возвращать массив (np.array) 3 на 3, где в каждой строке должны быть координаты одной ключевой точки. Порядок ключевых точек: head, wristL, wristR. Тип массива – np.float32.
Будет замеряться время работы этого метода, поэтому в нем не должно быть ничего лишнего (визуализации и т. п.).

Для каждого тест-кейса объект класса будет создаваться занова.

Пример скрипта (заглушка):

Файл **solver.py**

```
import cv2
import numpy as np
import time
import numpy.random

class Solver:

    def __init__(self):
        pass

    def process(self, data):

        depth, mask, rgb = data

        #do something
        time.sleep(0.5)

        #fill result
        head   = np.random.rand(3).astype(np.float32)
        wristL = np.random.rand(3).astype(np.float32)
        wristR = np.random.rand(3).astype(np.float32)

        return np.array([head, wristL, wristR], dtype=np.float32)
```