Лабораторная работа №2

Калибровка камеры и проекция 3D-координат в систему координат объекта

1. ЦЕЛЬ РАБОТЫ:

Целью данной лабораторной работы является изучение методов преобразования координат из трёхмерного (3D) пространства в двумерное (2D) изображение с использованием калибровочных параметров камеры. В рамках работы необходимо:

- Осуществить калибровку камеры и корректно считать её параметры (матрица K, коэффициенты дисторсии D, параметры вращения r и смещения t).
- Преобразовать 3D-координаты объектов (например, дорожной плоскости и виртуального 3D-ромба) в 2D-координаты изображения.
- Отобразить на видео виртуальные объекты: неизменные линии дорожки (плоскость земли, Z=0) и вращающийся 3D-ромб.

2. ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

2.1. Проекция 3D-точек в 2D-изображение

При съемке видео камера фиксирует двумерное изображение, однако реальная сцена существует в трёхмерном пространстве. Для того чтобы виртуальные объекты можно было корректно наложить на видео, необходимо преобразовать 3D-координаты объектов в 2D-координаты изображения. Это делается с помощью матрицы камеры, которая содержит внутренние параметры камеры (фокусное расстояние, координаты главной точки) и позволяет учитывать перспективные эффекты.

Матрица K имеет вид:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \quad (1)$$

где:

- f_x, f_y – фокусные расстояния (определяют «зум» или масштаб проекции);
- c_x, c_y – координаты главной точки (центр изображения).

При наличии 3D-точки P=(X,Y,Z) в мировой системе координат её проекция в пиксельные координаты (u, v) получается по формуле:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \cdot (R|t) \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \quad (2)$$

где R – матрица вращения, а t – вектор смещения (внешние параметры камеры).

Например:

Если вы знаете, что камера имеет f_x=800 пикселей и c_x=515, то точка, находящаяся в мировой системе, при проекции будет смещена таким образом, что горизонтальная координата изображения будет отклонена на 515 пикселей (это соответствует центру кадра).

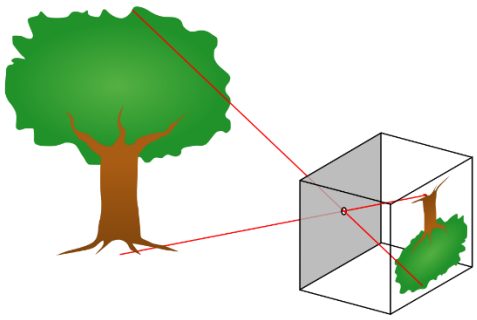

Рисунок 1 - Схема проекции 3D-точки в 2D-изображение

2.2. Калибровка камеры и YAML-файл

Процесс калибровки позволяет определить внутренние параметры камеры (матрица K) и коэффициенты дисторсии D, а также внешние параметры (поворот r и смещение t). Эти параметры необходимы для корректной проекции 3D-точек в 2D-пространство и устранения искажений, вызванных объективом.

```
%YAML:1.0
---
K: !!opencv-matrix # K – матрица камеры
   rows: 3
   cols: 3
   dt: d
   data: [ 6.9511361561000001e+02, 0., 4.8116952818999999e+02, 0.,
      6.9511361561000001e+02, 2.6555048459000000e+02, 0., 0., 1. ]
D: !!opencv-matrix # D – коэффициенты дисторсии
   rows: 5
   cols: 1
   dt: d
   data: [ -1.1983283309789111e-01, 2.7076763925130121e-01, 0., 0.,
      -7.3458604303021896e-02 ]
r: !!opencv-matrix # r и t – внешние параметры
   rows: 3
   cols: 1
   dt: d
   data: [ -9.9000000000000005e-02, 3.2999999999999662e-02,
      5.1490929575681577e-15 ]
t: !!opencv-matrix
   rows: 3
   cols: 1
   dt: d
   data: [ 1.7655262823707657e-01, -5.0000000000000000e-01,
      2.1051564943421215e+00 ]
sz: [ 960, 540 ] # sz – исходное разрешение, для которого проводилась
калибровка
remapped:
   []
dm: simple
```

При работе с видео важно, чтобы разрешение видео соответствовало указанному в YAML-файле, или чтобы матрица K была скорректирована под текущее разрешение.

2.3. Вращение 3D-объектов и кватернионы

Для динамического вращения 3D-объектов в пространстве можно использовать:

1. Матрицы вращения (где sin и cos явно применяются при построении матрицы 3×3).
2. Кватернионы, которые предоставляют альтернативный способ задания поворота без риска столкнуться с проблемами, связанными с углами Эйлера.

Кватернион q, описывающий поворот на угол θ вокруг единичной оси (v_x, v_y, v_z) задаётся формулой:

$$q = \cos\left(\frac{\theta}{2}\right) + sin\left(\frac{\theta}{2}\right) \cdot (v_x \boldsymbol{i} + v_y \boldsymbol{j} + v_z \boldsymbol{k}), \quad (3)$$

где i,j,k– базисные векторы по осям x,y,z.

Таким образом, cos и sin неявно участвуют в вычислении скалярной и векторной частей кватерниона.

Если мы хотим повернуть 3D-точку P=(x,y,z) вокруг заданной оси, то:

1. Интерпретируем P как «чисто векторный» кватернион p=(0,x,y,z).
2. Выполняем умножение:

$$p' = q \cdot p \cdot q^{-1}, \tag{4}$$

где q^{-1} – обратный (инверсный) кватернион.

3. В итоге получаем новый кватернион p', чьи компоненты (p'_x, p'_y, p'_z) задают координаты повернутой точки.

Пример реализации

В коде (рисунок 2) класс Quaternion содержит метод make_quaternion(angle, v_x, v_y, v_z), где sin и cos вычисляются внутри, а также методы для умножения кватернионов и их нормализации.

```
@staticmethod
def make_quaternion(angle, vx=0, vy=0, vz=1):
    return Quaternion(
        np.cos(angle / 2),
        vx * np.sin(angle / 2),
        vy * np.sin(angle / 2),
        vz * np.sin(angle / 2),
    )
```

Рисунок 2 – Пример реализации метода make_quaternion

Далее, при умножении векторов (рисунок 3):

```
for vector in radiusVectors:
    qVector = Quaternion(0, vector[0], vector[1], vector[2])
    final_quat = quat.multiply(qVector).multiply(quat.inversed())
    rotatedVector = [final_quat.vx(),
                     final_quat.vy(),
                     final_quat.vz()]
```

Рисунок 3 – Пример реализации класса Quaternion

sin и cos уже «спрятаны» внутри кватерниона quat, поворот происходит за счёт операций умножения кватернионов. При отрисовке 3D-ромба боковые точки вращаются вокруг некоторого центра, вектор от центра до точки интерпретируется как кватернион, умножается на поворотный кватернион, а затем преобразуется обратно в координаты.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ И ПРАКТИЧЕСКАЯ ЧАСТЬ

1. Скачайте архив с данными (содержащий видеофайлы в формате AVI и калибровочный файл leftImage.yml) по ссылке: https://cloud.mail.ru/public/abmx/Li8kWSVrU.
2. Снимите своё видео с нужной сценой (например, дорожку или улицу), которое будет использоваться в качестве фона для наложения виртуальных объектов. Сохраните видео в формате .avi (в папку data/city или укажите свои пути к видео).
3. Настройте значения в YAML-файле так, чтобы они соответствовали вашим данным.
4. Склонируйте репозиторий cv_book.
5. Объедините код отрисовки дорожки (task_1) и вращающегося объекта (task_4) в один обработчик (например, класс Reader, наследуемый от SeasonReader).
6. Запустите программу и проверьте, как на видео накладываются виртуальные объекты.
7. Скорректируйте параметры (shift_x, shift_y, shift_z, размеры и т.д.) для достижения оптимального результата.
8. Полученный результат направьте преподавателю на проверку.

Задача 1 (Module_1a, task_1):

Сделать класс для отрисовки на земле (Z=0) линий в системе координат объекта.

Пакет spatial_geometry_tools:

- calib.Calib – класс параметров камеры, который считывает матрицу K (интринсики), коэффициенты дисторсии D, вектор поворота r и смещение t из YAML-файла.
- camera.Camera – класс для перехода из 3D в 2D. Для проекции точек используется метод project_point_3d_to_2d.
- point.Point3d – класс для представления 3D-точки с координатами (x, y, z).

- **Дано:**

- Параметры камеры (из YAML-файла), стандартная ширина пути ≈1.6 м (то есть для построения линии по оси X от центра откладываются по 0.8 м влево и вправо).
- Длина путей (настраиваемый параметр в метрах).

- **Решение:**
  - Создается класс WayEstimator, который принимает:
    - calib_dict: dict – словарь параметров камеры, считанный из YAML-файла.
    - ways_length: int – длина пути (в метрах).
  - Для преобразования координат из 3D в 2D используются матрицы аффинных преобразований, построенные на основе параметров камеры.
  - Отрисовка линий производится с помощью OpenCV (функция cv2.line).

Задача 4 (Module_1a, task_4):
Реализация вращения виртуального объекта с использованием кватернионов.

Применяется класс Quaternion и QuaternionRotate для поворота объектов (например, кубов или ромбов). В коде показано, как вычисляется поворот, используя функцию np.cos(angle/2) и np.sin(angle/2), и как осуществляется умножение кватернионов для получения конечного результата.

В дополнение к главное задаче можно рассмотреть task_2, task_3, task_5, task_6, task_7:

- ❖ Сделать класс для отрисовки эмблемы Университета МИСИС и выводе данных о скорости движения объекта и его координатах B, L.
- ❖ Сделать класс для преобразования изображение в BirdsView заданных размеров (подкорректировать точность - вертикальность).
- ❖ Сделать класс для отображения плоскости дороги цветом, а потом в виде маски позволяющей скрыть всю информацию на изображении кроме дороги.
- ❖ Сделать класс для отображения плоскости дороги в виде сетки с заданным шагом и длиной и шириной.

Литература:

- Документация OpenCV – https://opencv.org/

- Учебное пособие по калибровке камеры в OpenCV - https://waksoft.susu.ru/2020/02/29/kalibrovka-kamery-s-ispolzovaniem-s-opencv/
- Ресурсы по работе с кватернионами – https://quaternions.online/
- Дополнительная шпаргалка по компьютерному зрению – https://tproger.ru/translations/opencv-python-guide

ПРИЛОЖЕНИЕ

Пример итогового результата: https://disk.yandex.ru/i/2o_KV8od6-4Shw