Санкт-Петербургский политехнический университет Петра Великого
Институт машиностроения, материалов и транспорта
Высшая школа автоматизации и робототехники

Отчёт

по лабораторной работе №1

Дисциплина: Техническое зрение

Тема: Моделирование движения робота с использованием библиотеки OpenCV

Студент гр. 3331506/70401 Козлов Д. А.

Преподаватель Варлашин В. В.

« »_____2020 г.

Санкт-Петербург

2020

Задание

Пользуясь библиотекой OpenCV, разработать графический симулятор управления движением робота. Симулятор должен выполнять следующие задачи:

1. Отрисовка условной модели робота на фоновом изображении.
2. Движение робота вперед и назад, влево и вправо.
3. Поворот робота по часовой стрелке и против относительно центра его корпуса.
4. Невыезд робота за область передвижения.

Ход работы

Общее описание реализации

Для описания модели робота был создан класс *MyRobot*, поля которого представлены на рисунке 1.

```
private:
    Point2f m_center;
    float m_angle;
    float m_width;
    float m_height;
    float m_wheelWidth;
    float m_wheelDiameter;
    float m_speed;
    float m_angularSpeed;
    Size2i m_area;
```

Рисунок 1 – Поля класса *MyRobot*

Общий алгоритм работы симулятора:

1) Отрисовка робота в начальном центральном положении.

2) Ожидание нажатия клавиши.

3) В зависимости от нажатой клавиши:

- Esc – завершение программы;
- w, a, s, или d – движение робота вперёд, влево, назад или вправо соответственно со скоростью *m_speed* пикселей за 1 нажатие;

- q или e – поворот робота против часовой стрелки или по часовой стрелке соответственно со скоростью *m_angularSpeed* радиан за 1 нажатие.

4) Если координаты робота не выходят за область передвижений (*m_area*), то далее п. 5, иначе – п. 6
5) Отрисовка робота в новом положении, далее п. 2.
6) Возвращение координат робота к тем, которые он занимал до выезда за границу области передвижений, далее п. 2.

1. Отрисовка робота (метод *draw*)

В качестве параметров функция принимает 2 *Mat*-объекта: *backgroundImage* (фон) и *robotImage* (изображение, в которое будет скопирован фон, на котором нарисуется робот).

Изначально координаты точек робота задаются в локальной системе, расположение которой изображено на рисунке 2.

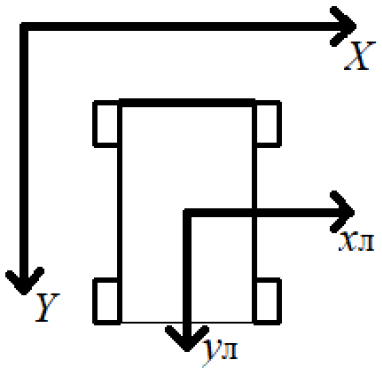

Рисунок 2 – Расположение локальной системы координат

Затем локальные координаты приводятся к глобальным путём вызова функции *localToGlobal(Point2f pt)*, которая умножает матрицу поворота на матрицу координат точек робота в локальной системе:

$$\begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} = \begin{pmatrix} cos\,\alpha & -sin\,\alpha & tx \\ sin\,\alpha & cos\,\alpha & ty \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} xл \\ yл \\ 1 \end{pmatrix},$$

где α – угол поворота робота, tx и ty – координаты центра робота в глобальной системе координат (*m_center*).

Далее точки соединяются при помощи функции *line()* библиотеки OpenCV.

2. Движение робота (метод *move*)

Данная функция параметром принимает направление движения, а затем с учетом угла наклона робота изменяет координаты центра. Код функции представлен на рисунке 3.

```
void MyRobot::move(char direction)
{
    if (direction == 'u')
    {
        setCenter(m_center.x + (m_speed * sin(m_angle)), m_center.y - (m_speed * cos(m_angle)));
    }
    if (direction == 'd')
    {
        setCenter(m_center.x - (m_speed * sin(m_angle)), m_center.y + (m_speed * cos(m_angle)));
    }
    if (direction == 'l')
    {
        setCenter(m_center.x - (m_speed * cos(m_angle)), m_center.y - (m_speed * sin(m_angle)));
    }
    if (direction == 'r')
    {
        setCenter(m_center.x + (m_speed * cos(m_angle)), m_center.y + (m_speed * sin(m_angle)));
    }
}
```

Рисунок 3 – Реализация движения робота

3. Поворот робота (метод rotate)

Данная функция прибавляет к текущему значению угла поворота робота значение, которая функция принимает в качестве параметра. Реализация метода приведена на рисунке 4.

```
void MyRobot::rotate(float angle)
{
    m_angle += angle;
}
```

Рисунок 4 – Реализация поворота робота

4. Невыезд за область передвижения

Невыезд робота осуществляется благодаря функции *isInArea(Point2f pt)*, которая проверяет точку на принадлежность области передвижения. Код функции представлен на рисунке 5.

```
bool MyRobot::isInArea(Point2f pt)
{
    if (pt.x > m_area.width || pt.x  < 0 ||
        pt.y > m_area.height || pt.y < 0)
    {
        return false;
    }
    return true;
}
```

Рисунок 5 – функция *isInArea*

После нажатия пользователем какой-либо управляющей кнопки, перед началом движения координаты робота (центр и угол поворота) сохраняются в буферные переменные, затем при вызове функции *draw* происходит проверка четырех крайних угловых точек робота функцией *isInArea*. Если проверка не пройдена, то робот не рисуется и его координаты возвращаются к сохраненным ранее буферным значениям. Таким образом, робот не рисуется за границами области передвижения и фактические координаты робота находятся внутри нее.

Вывод

В результате выполнения лабораторной работы был реализован простейший графический симулятор управления роботом. Получены первоначальные навыки работы со средствами библиотеки OpenCV.