Санкт-Петербургский политехнический университет Петра Великого
Институт металлургии, машиностроения и транспорта
Кафедра компьютерных технологий в машиностроении

Лабораторная работа №1

По дисциплине: Техническое зрение

Студент гр. 3331506/70401 Кочурин Р.П.
Преподаватель Варлашин В.В.

« »____2020 г.

Санкт-Петербург
2020

Цель работы

Приобрести первоначальные навыки работы с библиотеками OpenCV.

Задание

Реализовать алгоритм управления роботом с клавиатуры на некотором пространстве с запретом выхода за границу.

Выполнение

На рисунке 1 представлены переменные класса *MyRobot*.

```
float m_speedAlong;
float m_speedAcross;
float m_angularSpeed;
Point2f m_center;
float m_angle;
const float m_width;
const float m_height;
const float m_wheelWidth;
const float m_wheelDiameter;
Size2i m_area;

//точки робота
//лобовая сторона смотрит вверх
struct rectangle m_body;
struct rectangle m_wheel[4];    //0 - UpLeft;       1 - UpRight;
                                //2 - DownLeft;     3 - DownRight
```

Рисунок 1 – Переменные класса MyRobot

На рисунке 2 представлена функция управления движением робота с клавиатуры.

```
    //основная функция управления с клавиатуры
int MyRobot::m_Motion(Mat &image)
{
    while (waitKey(10) != 27)
    {
        int SPEED = 10;
        int along = 0;
        int across = 0;
        float angular = 0;
        char key = waitKey(30);
        if ((key == 'w') || (key == 'W'))
            along = SPEED;
        if ((key == 's') || (key == 'S'))
            along = -SPEED;
        if ((key == 'd') || (key == 'D'))
            across = SPEED;
        if ((key == 'a') || (key == 'A'))
            across = -SPEED;
        if ((key == 'q') || (key == 'Q'))
            angular = SPEED  * PI / 180;
        if ((key == 'e') || (key == 'E'))
            angular = -SPEED * PI / 180;
        if (key == 32)
            m_Zeroing();
        m_Move(along, across);
        m_Rotate(angular);
        m_Positioning();
        m_Drawing(image);
    }
    return 0;
```

Рисунок 2 – Функция управления роботом с клавиатуры

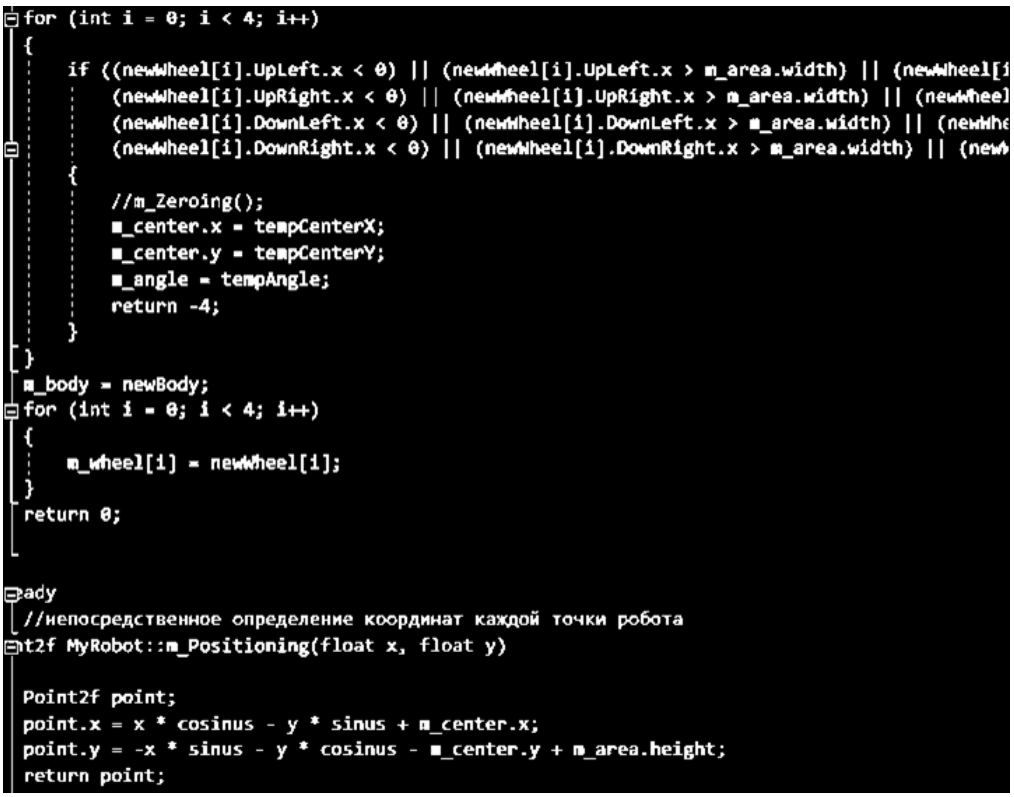
Цикл *while* этой функции будет выполняться пока на клавиатуре не нажать *Esc*. С функция *m_Move* задаёт продольную и поперечную скорость робота, функция *m_Rotate* – круговую скорость, функция *m_Positioning*

производит перерасчёт положения всех точек робота, а функция *m_Drowing* производит вывод робота на экран пользователя.

На рисунках 3 и 4 представлена реализация функции *m_Positioning*, а также на рисунке 4 присутствует реализация вспомогательной функции *m_Positioning* с двумя параметрами, которые задают смещение по осям *x* и *y* в относительной системе координат, связанной с центром робота.

```
MyRobot::m_Positioning()

//копирование на случай врезания в рамку
float tempCenterX = m_center.x;
float tempCenterY = m_center.y;
float tempAngle = m_angle;

sinus = sin(m_angle);
cosinus = cos(m_angle);

m_center.x = m_center.x + getSpeedAcross() * cosinus - getSpeedAlong() * sinus;
m_center.y = m_center.y + getSpeedAlong() * cosinus + getSpeedAcross() * sinus;
m_angle = m_angle + getAngularSpeed();

struct rectangle newBody;
newBody.UpLeft = m_Positioning(-m_width / 2, m_height / 2);
newBody.UpRight = m_Positioning(m_width / 2, m_height / 2);
newBody.DownLeft = m_Positioning(-m_width / 2, -m_height / 2);
newBody.DownRight = m_Positioning(m_width / 2, -m_height / 2);
struct rectangle newWheel[4];   //0 - UpLeft;       1 - UpRight;
                                //2 - DownLeft;     3- DownRight
//0 - UpLeft
newWheel[0].UpLeft = m_Positioning(-m_width / 2 - m_wheelWidth, m_height / 2);
newWheel[0].UpRight = newBody.UpLeft;
newWheel[0].DownLeft = m_Positioning(-m_width / 2 - m_wheelWidth, m_height / 2 - m_wheelDi
newWheel[0].DownRight = m_Positioning(-m_width / 2, m_height / 2 - m_wheelDiameter);
//1 - UpRight
newWheel[1].UpLeft = newBody.UpRight;
newWheel[1].UpRight = m_Positioning(m_width / 2 + m_wheelWidth, m_height / 2);
newWheel[1].DownLeft = m_Positioning(m_width / 2 , m_height / 2 - m_wheelDiameter);
newWheel[1].DownRight = m_Positioning(m_width / 2 + m_wheelWidth, m_height / 2 - m_wheelDi
//2 - DownLeft
newWheel[2].UpLeft = m_Positioning(-m_width / 2 - m_wheelWidth, -m_height / 2 + m_wheelDia
newWheel[2].UpRight = m_Positioning(-m_width / 2, -m_height / 2 + m_wheelDiameter);
newWheel[2].DownLeft = m_Positioning(-m_width / 2 - m_wheelWidth, -m_height / 2);
newWheel[2].DownRight = newBody.DownLeft;
//3- DownRight
newWheel[3].UpLeft = m_Positioning(m_width / 2, -m_height / 2 + m_wheelDiameter);
newWheel[3].UpRight = m_Positioning(m_width / 2 + m_wheelWidth, -m_height / 2 + m_wheelDia
newWheel[3].DownLeft = newBody.DownRight;
newWheel[3].DownRight = m_Positioning(m_width / 2 + m_wheelWidth, -m_height / 2);

//проверка на корректность координат
if ((newBody.UpLeft.x < 0) || (newBody.UpLeft.x > m_area.width) || (newBody.UpLeft.y < 0)
    (newBody.UpRight.x < 0) || (newBody.UpRight.x > m_area.width) || (newBody.UpRight.y <
    (newBody.DownLeft.x < 0) || (newBody.DownLeft.x > m_area.width) || (newBody.DownLeft.y
    (newBody.DownRight.x < 0) || (newBody.DownRight.x > m_area.width) || (newBody.DownRigh
{
    //m_Zeroing();
    m_center.x = tempCenterX;
    m_center.y = tempCenterY;
    m_angle = tempAngle;
    return -4;
}
```

Рисунок 3 – Реализация функции *m_Positioning*



```
for (int i = 0; i < 4; i++)
{
    if ((newWheel[i].UpLeft.x < 0) || (newWheel[i].UpLeft.x > m_area.width) || (newWheel[i
        (newWheel[i].UpRight.x < 0) || (newWheel[i].UpRight.x > m_area.width) || (newWheel
        (newWheel[i].DownLeft.x < 0) || (newWheel[i].DownLeft.x > m_area.width) || (newWhe
        (newWheel[i].DownRight.x < 0) || (newWheel[i].DownRight.x > m_area.width) || (new
    {
        //m_Zeroing();
        m_center.x = tempCenterX;
        m_center.y = tempCenterY;
        m_angle = tempAngle;
        return -4;
    }
}
m_body = newBody;
for (int i = 0; i < 4; i++)
{
    m_wheel[i] = newWheel[i];
}
return 0;


ady
//непосредственное определение координат каждой точки робота
t2f MyRobot::m_Positioning(float x, float y)

Point2f point;
point.x = x * cosinus - y * sinus + m_center.x;
point.y = -x * sinus - y * cosinus - m_center.y + m_area.height;
return point;
```

Рисунок 4 – Реализация функции *m_Positioning* и вспомогательной функции

Данная функция работает следующим образом:

1) Запоминает старые значения координат центра робота и угла поворота.
2) Происходит пересчёт синусов и косинусов угла поворота.
3) Пересчёт координат центра робота.
4) Пересчёт угла поворота робота.
5) С помощью вспомогательной функции пересчитываем координаты каждой точки робота.

Так как ранее было сказано про относительные координаты, то на рисунке 5 представлены системы координат, использованные при выводе формул для координат точек.

Рисунок 5 – Использованные в работе системы координат

Далее приведены использованные матрицы перехода и формулы по переходу из относительных координат, связанных с роботом, к глобальным координатам, связанных с началом координат рисунка.

$$H_{01} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & h \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix};$$

$$H_{12} = \begin{pmatrix} \cos q_3 & -\sin q_3 & 0 & q_1 \\ \sin q_3 & \cos q_3 & 0 & q_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix};$$

$$R_M^{(2)} = \begin{pmatrix} a \\ b \\ 0 \end{pmatrix}; \quad R_M^{(0)} = \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} = H_{01} H_{12} R_M^{(2)}.$$

В данных формулах использованы:

h – высота рисунка;

q_1 – координата центра системы координат 3 по координате x в системе координат 2;

q_2 – координата центра системы координат 3 по координате y в системе координат 2;

q_3 – угловой поворот третьей системы координат относительно второй системы координат;

a и b – координаты x и y соответственно в относительных координатах, связанных с центром робота.

В итоге получились следующие формулы для точек робота, которые использовались во вспомогательной функции:

$$x = a\cos q_3 - b\sin q_3 + q_1;$$
$$y = -a\sin q_3 - b\cos q_3 - q_2 + h.$$

Вывод

В ходе выполнения работы были приобретены первичные навыки работы с библиотеками OpenCV, в следствии чего был реализована поставленная задача.