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 Якименко Г.К.

Преподаватель Варлашин В.В.

« »_____2020 г.

Санкт-Петербург

2020

my_robot.h

```
#pragma once
#include "opencv2/core.hpp"
#include "opencv2/imgproc.hpp"
#include "opencv2/highgui.hpp"
#include <time.h>

class MyRobot
{
private:
    //Параметры робота
    cv::Point2f m_centr; //Координаты центра
    float m_width; //Ширина
    float m_height; //Высота
    float m_wheelWidth; //Ширина колёс
    float m_wheelDiometr; //Диаметр колёс
    float m_speed; //Скорость робота
    float m_angularSpeed; //Угловая скорость робота
    float m_angle; //Угол поворота робота
    cv::Size2i m_area;
    cv::Point m_p[16]; //Координаты точек
    cv::Mat m_image;
    float memory[3]; //0,1 - координаты центра. 2 - угол поворота
public:
    //Конструктор
    MyRobot();
    MyRobot(float width, float height,
        float wheelWidth, float wheelDiometr,
        float speed, float angularSpeed,
        cv::Size2i area);
    //Деструктор
    ~MyRobot();
    //Обновление координат робота
    //относительно центра робота
    void updateCoord();
    //Движение робота по клавишам
    void move();
    //Поворот точек робота относительно
    //его центра
    void rotate();
    //Отрисовка робота
    void drawRobot();
    //Закрашивание робота
    void clearRobot();
    //Проверка границ
    void checkArea();
};
```

my_robot.cpp

```
#pragma once
#include "../OpenCV_test/my_robot.h"
#define PI2 6.28318

MyRobot::MyRobot(float width, float height,
                 float wheelWidth, float wheelDiometr,
                 float speed, float angularSpeed,
                 cv::Size2i area) :
            m_width(width),
            m_height(height),
            m_wheelWidth(wheelWidth),
            m_wheelDiometr(wheelDiometr),
            m_speed(speed),
            m_angularSpeed(angularSpeed),
            m_area(area)
{
    //Параметры поля
    m_image = { m_area, CV_8UC3, cv::Scalar(255, 255, 255) };

    //Координаты центра робота = координаты центра поля
    m_centr.x = (float)m_area.width / 2;
    m_centr.y = (float)m_area.height / 2;

    //Угол поворота робота ПУ 0
    m_angle = 0;

    //Обновляем координаты точек робота
    //относительно центра
    updateCoord();

    //Отрисовка робота
    drawRobot();
}

MyRobot::~MyRobot()
{
    ;
}

void MyRobot::updateCoord()
{
/*
Точки для линий
0----------------------> X
|
|      p4--p0-----p1--p7
|       |  |       |  |
|      p5--p6     p9--p8
|          |       |
|          |       |
|     p10--p12   p15--p13
```

```
|       |  |       |  |
|     p11--p3-----p2--p14
|
v
Y
*/
    //Расчет координат прямоугольника
    //относительно центра робота
    m_p[0].x = m_centr.x - (m_width / 2);
    m_p[0].y = m_centr.y - (m_height / 2);

    m_p[1].x = m_centr.x + (m_width / 2);
    m_p[1].y = m_centr.y - (m_height / 2);

    m_p[2].x = m_centr.x + (m_width / 2);
    m_p[2].y = m_centr.y + (m_height / 2);

    m_p[3].x = m_centr.x - (m_width / 2);
    m_p[3].y = m_centr.y + (m_height / 2);

    //Расчет координат колёс
    //относительно центра робота
    m_p[4].x = m_p[0].x - m_wheelWidth;
    m_p[4].y = m_p[0].y;

    m_p[5].x = m_p[4].x;
    m_p[5].y = m_p[4].y + m_wheelDiometr;

    m_p[6].x = m_p[0].x;
    m_p[6].y = m_p[5].y;

    m_p[7].x = m_p[1].x + m_wheelWidth;
    m_p[7].y = m_p[0].y;

    m_p[8].x = m_p[7].x;
    m_p[8].y = m_p[4].y + m_wheelDiometr;

    m_p[9].x = m_p[1].x;
    m_p[9].y = m_p[4].y + m_wheelDiometr;

    m_p[10].x = m_p[0].x - m_wheelWidth;
    m_p[10].y = m_p[3].y - m_wheelDiometr;

    m_p[11].x = m_p[0].x - m_wheelWidth;
    m_p[11].y = m_p[3].y;

    m_p[12].x = m_p[0].x;
    m_p[12].y = m_p[10].y;

    m_p[13].x = m_p[7].x;
    m_p[13].y = m_p[10].y;

    m_p[14].x = m_p[7].x;
```

```
    m_p[14].y = m_p[3].y;

    m_p[15].x = m_p[1].x;
    m_p[15].y = m_p[10].y;

    //Поворот точек на нужный угол
    rotate();
}

void MyRobot::clearRobot()
{
    //Цвет и ширина линий прямоугольника
    cv::Scalar lineColor(255, 255, 255);
    int lineHeight = 1;
    /*
Точки для линий
0----------------------> X
|
|      p4--p0-----p1--p7
|       |  |       |  |
|      p5--p6     p9--p8
|          |       |
|          |       |
|     p10--p12   p15--p13
|       |  |       |  |
|     p11--p3-----p2--p14
|
v
Y
*/

     //Отрисовка линий прямоугольника
    cv::line(m_image, m_p[0], m_p[1], lineColor, lineHeight, 8,
0);
    cv::line(m_image, m_p[1], m_p[2], lineColor, lineHeight, 8,
0);
    cv::line(m_image, m_p[2], m_p[3], lineColor, lineHeight, 8,
0);
    cv::line(m_image, m_p[3], m_p[0], lineColor, lineHeight, 8,
0);

    //Отрисовка колёс
    cv::line(m_image, m_p[0], m_p[4], lineColor, lineHeight, 8,
0);
    cv::line(m_image, m_p[4], m_p[5], lineColor, lineHeight, 8,
0);
    cv::line(m_image, m_p[5], m_p[6], lineColor, lineHeight, 8,
0);

    cv::line(m_image, m_p[1], m_p[7], lineColor, lineHeight, 8,
0);
    cv::line(m_image, m_p[7], m_p[8], lineColor, lineHeight, 8,
0);
```

```
    cv::line(m_image, m_p[8], m_p[9], lineColor, lineHeight, 8,
0);

    cv::line(m_image, m_p[12], m_p[10], lineColor, lineHeight, 8,
0);
    cv::line(m_image, m_p[10], m_p[11], lineColor, lineHeight, 8,
0);
    cv::line(m_image, m_p[11], m_p[3], lineColor, lineHeight, 8,
0);

    cv::line(m_image, m_p[15], m_p[13], lineColor, lineHeight, 8,
0);
    cv::line(m_image, m_p[13], m_p[14], lineColor, lineHeight, 8,
0);
    cv::line(m_image, m_p[14], m_p[2], lineColor, lineHeight, 8,
0);

    cv::line(m_image, m_p[6], m_centr, lineColor, lineHeight, 8,
0);
    cv::line(m_image, m_p[9], m_centr, lineColor, lineHeight, 8,
0);


}

void MyRobot::drawRobot()
{
    //Цвет и ширина линий прямоугольника
    cv::Scalar lineColor(0, 0, 0);
    int lineHeight = 1;

    /*
Точки для линий
0----------------------> X
|
|      p4--p0-----p1--p7
|       |  |       |  |
|      p5--p6     p9--p8
|          |       |
|          |       |
|     p10--p12   p15--p13
|       |  |       |  |
|     p11--p3-----p2--p14
|
v
Y
*/

  //Отрисовка линий прямоугольника
    cv::line(m_image, m_p[0], m_p[1], lineColor, lineHeight, 8,
0);
    cv::line(m_image, m_p[1], m_p[2], lineColor, lineHeight, 8,
0);
```

```
    cv::line(m_image, m_p[2], m_p[3], lineColor, lineHeight, 8,
0);
    cv::line(m_image, m_p[3], m_p[0], lineColor, lineHeight, 8,
0);

    //Отрисовка колёс
    cv::line(m_image, m_p[0], m_p[4], lineColor, lineHeight, 8,
0);
    cv::line(m_image, m_p[4], m_p[5], lineColor, lineHeight, 8,
0);
    cv::line(m_image, m_p[5], m_p[6], lineColor, lineHeight, 8,
0);

    cv::line(m_image, m_p[1], m_p[7], lineColor, lineHeight, 8,
0);
    cv::line(m_image, m_p[7], m_p[8], lineColor, lineHeight, 8,
0);
    cv::line(m_image, m_p[8], m_p[9], lineColor, lineHeight, 8,
0);

    cv::line(m_image, m_p[12], m_p[10], lineColor, lineHeight, 8,
0);
    cv::line(m_image, m_p[10], m_p[11], lineColor, lineHeight, 8,
0);
    cv::line(m_image, m_p[11], m_p[3], lineColor, lineHeight, 8,
0);

    cv::line(m_image, m_p[15], m_p[13], lineColor, lineHeight, 8,
0);
    cv::line(m_image, m_p[13], m_p[14], lineColor, lineHeight, 8,
0);
    cv::line(m_image, m_p[14], m_p[2], lineColor, lineHeight, 8,
0);

    cv::line(m_image, m_p[6], m_centr, lineColor, lineHeight, 8,
0);
    cv::line(m_image, m_p[9], m_centr, lineColor, lineHeight, 8,
0);



    cv::imshow("robot", m_image);
}

void MyRobot::rotate()
{
    for (int i = 0; i < 16; i++) {
        //Координаты точки относительно СК робота
        float dx, dy;
        //Перевод точки в СК робота
        dx = m_p[i].x - m_centr.x;
        dy = m_p[i].y - m_centr.y;
        //Поворот точки относительно СК робота
```

```
        //и перевод координат точки обратно в глобальную СК
        m_p[i].x = (dx * cosf(m_angle * 6.28318 / 360) - dy *
sinf(m_angle * PI2 / 360)) + m_centr.x;
        m_p[i].y = (dx * sinf(m_angle * 6.28318 / 360) + dy *
cosf(m_angle * PI2 / 360)) + m_centr.y;
    }
}

void MyRobot::checkArea()
{
    //Перебираем самые удаленные от центра точки: p4, p7, p11, p14
    for (int i = 4; i < 15; )
    {
        //Проверяем пересекает ли точка границы по X
        if (m_p[i].x >= m_area.width)
        {
            //Сдвигаем координату центра так, чтобы
            //крайняя точка не выходила за границы
            m_centr.x -= (m_p[i].x - m_area.width);
            //Обновляем координаты всех точек
            updateCoord();
        }
        if (m_p[i].x <= 0 )
        {
            //Сдвигаем координату центра так, чтобы
            //крайняя точка не выходила за границы
            m_centr.x += abs(m_p[i].x);
            //Обновляем координаты всех точек
            updateCoord();
        }

        //Проверяем пересекает ли точка границы по Y
        if (m_p[i].y >= m_area.height)
        {
            //Сдвигаем координату центра так, чтобы
            //крайняя точка не выходила за границы
            m_centr.y -= (m_p[i].y - m_area.height);
            //Обновляем координаты всех точек
            updateCoord();
        }
        if (m_p[i].y <= 0)
        {
            //Сдвигаем координату центра так, чтобы
            //крайняя точка не выходила за границы
            m_centr.y += abs(m_p[i].y);
            //Обновляем координаты всех точек
            updateCoord();
        }

        // Обновление счетчика i
        switch (i)
        {
        case 4:
```

```
                i = 7;
                break;
            case 7:
                i = 11;
                break;
            case 11:
                i = 14;
                break;
            case 14:
                i = 15;
                break;
            }
        }
    }

    void MyRobot::move()
    {
        switch (cv::waitKey(1))
        {
        //Вперед
        case 'w':
            //Стираем предыдущее изображение робота
            clearRobot();
            //Перемещаем центр робота в нужном направлении
            m_centr.y -= m_speed * cosf(m_angle * PI2 / 360);
            m_centr.x += m_speed * sinf(m_angle * PI2 / 360);
            //Обновление всех координат робота
            updateCoord();
            //Проверяем не выходит ли робот за границы изображения
            checkArea();
            //Отрисовка модели робота на новой позиции
            drawRobot();
            break;
        //Назад
        case 's':
            //Стираем предыдущее изображение робота
            clearRobot();
            //Перемещаем центр робота в нужном направлении
            m_centr.y += m_speed * cosf(m_angle * PI2 / 360);
            m_centr.x -= m_speed * sinf(m_angle * PI2 / 360);
            //Обновление всех координат робота
            updateCoord();
            //Проверяем не выходит ли робот за границы изображения
            checkArea();
            //Отрисовка модели робота на новой позиции
            drawRobot();
            break;
        //Влево
        case 'a':
            //Стираем предыдущее изображение робота
            clearRobot();
            //Перемещаем центр робота в нужном направлении
            m_centr.x -= m_speed * cosf(m_angle * PI2 / 360);
```

```
            m_centr.y -= m_speed * sinf(m_angle * PI2 / 360);
            //Обновление всех координат робота
            updateCoord();
            //Проверяем не выходит ли робот за границы изображения
            checkArea();
            //Отрисовка модели робота на новой позиции
            drawRobot();
            break;
        //Вправо
        case 'd':
            //Стираем предыдущее изображение робота
            clearRobot();
            //Перемещаем центр робота в нужном направлении
            m_centr.x += m_speed * cosf(m_angle * PI2 / 360);
            m_centr.y += m_speed * sinf(m_angle * PI2 / 360);
            //Обновление всех координат робота
            updateCoord();
            //Проверяем не выходит ли робот за границы изображения
            checkArea();
            //Отрисовка модели робота на новой позиции
            drawRobot();
            break;
        //Поворот по часовой
        case 'e':
            //Стираем предыдущее изображение робота
            clearRobot();
            //Поворот
            m_angle += m_angularSpeed;
            if (m_angle >= 360) m_angle = 0;
            //Обновление всех координат робота
            updateCoord();
            //Проверяем не выходит ли робот за границы изображения
            checkArea();
            //Отрисовка модели робота на новой позиции
            drawRobot();
            break;
        //Поворот против часовой
        case 'q':
            //Стираем предыдущее изображение робота
            clearRobot();
            //Поворот
            m_angle -= m_angularSpeed;
            if (m_angle < 0) m_angle = 360 - m_angularSpeed;
            //Обновление всех координат робота
            updateCoord();
            //Проверяем не выходит ли робот за границы изображения
            checkArea();
            //Отрисовка модели робота на новой позиции
            drawRobot();
            break;
        }
}
```

main.cpp

```
#include <iostream>
#include "opencv2/core.hpp"
#include "opencv2/imgproc.hpp"
#include "opencv2/highgui.hpp"
#include "../OpenCV_test/my_robot.h"

using namespace cv;
using namespace std;

int main()
{
    //Желаемые хар-ки
    float width = 40; //Ширина
    float height = 70; //Высота
    float wheelWidth = 20; //Ширина колёс
    float wheelDiometr = 15; //Диаметр колёс
    float speed = 5; //Скорость робота
    float angularSpeed = 5; //Угловая скорость робота
    Size2i area_size(640, 480);


    //Экземпляр класса
    MyRobot robot(width, height, wheelWidth, wheelDiometr, speed,
angularSpeed, area_size);

    while (1)
    {
        robot.move();
    }
}
```