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 Самарин А.С.

Преподаватель Варлашин В.В.

« »\_\_\_\_\_\_\_\_2020 г.

Санкт-Петербург

2020

**Задание**

Методами OpenCV реализовать движение и поворот робота.

**Алгоритм работы**

1. Сканирование нажатых кнопок;
2. Расчет положения центра робота в зависимости от нажатой кнопки;
3. Создание и поворот всех точек корпуса робота;
4. Соединение точек линиями и вывод изображения.

**Описание робота**

Для создания робота был создан класс *MyRobot.* Данный класс обладает следующими параметрами:

Point2f m\_center; //к-ты центра прямоугольника

Size2i m\_barrier; //размеры барьера вокруг робота

float m\_width; //ширина корпуса

float m\_height; //высота корпуса

float m\_widthTower; //ширина башни

float m\_heightTower; //высота башни

float m\_wheelWidth; //ширина колес

float m\_wheelDiameter; //диаметр колес

float m\_speed; //скорость

float m\_angularSpeed; //скорость поворота

float m\_angle; //угол поворота робота

float m\_angleTower; //угол поворота башни

Size2i m\_area; //размер создаваемого поля

При создании класса вызывается конструктор, в котором заполняются все параметры робота.

**Сканирование кнопок**

Сканирование кнопок происходит в цикле *while* до момента пока пользователь не нажмет кнопку с кодом 27(*esc*).

while (waitKey(25) != 27)

{

key = waitKey(30);

//каждый раз создаем новое поля для отображения

Mat backgroundAndRobot(size, CV\_8UC3, white);

if ((key == 'w') || (key == 'W'))

{

robot.move(1);

}

Каждый раз заходя в цикл создается новый задний фон с помощью команды:

Mat backgroundAndRobot(size, CV\_8UC3, white);

Задний фон представляет из себя матрицу размером size, заполненную пикселями белого цвета.

Scalar white(255, 255, 255);

Size size(1600, 900);

Далее начинается сканирование кнопок и, в зависимости от нажатой кнопки вызывается или функция *move()* или *rotate()*, движение и поворот соответственно.

**Расчет положения центра робота**

Для перемещения центра робота используются две функции: *move()* и *checkPosition()*. Первая перемещает центр в зависимости от нажатой кнопки, вторая проверяет возможно ли движение в заданном направлении.

void MyRobot::move(int8\_t buttonNumber)

{

center = m\_center;

if ((buttonNumber == 1))

{

m\_center.x = m\_center.x + m\_speed \* sin(m\_angle);

m\_center.y = m\_center.y - m\_speed \* cos(m\_angle);

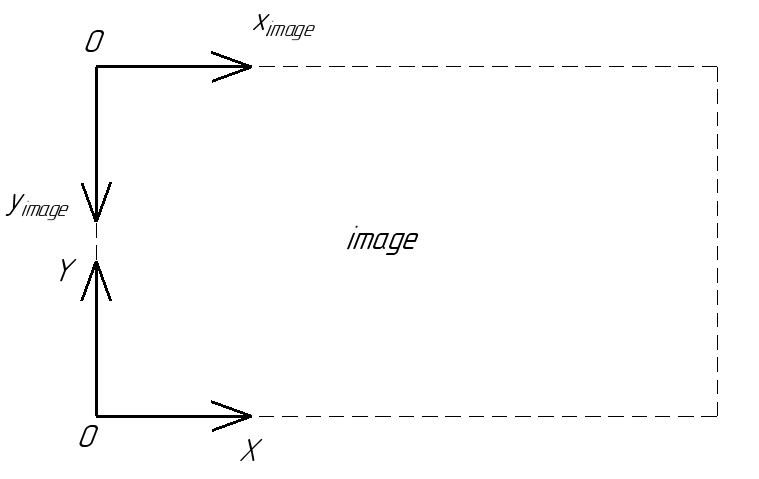
checkPosition(m\_center);

return;

}

В данной части функции представлено движение робота вверх.

Глобальная система координат имеет ноль в левом нижнем углу, а у изображения ноль находится в левом верхнем углу. Поэтому, для соответствия движения в разных системах координат, координата *y* при движении вверх отнимается. Благодаря чему робот на экране движется вверх.



Для реализации движения по диагоналям, если угол поворота отличается от нуля, скорость умножается на *sin* и *cos.*

Функция *checkPosition()* используется для проверки столкновения робота с границами. Идет проверка того, допустимы ли новые значения координат центра. Если недопустимы, то просто приравниваем их к предыдущим значениям.

int8\_t MyRobot::checkPosition(Point2f point)

{

if ((point.x > (m\_area.width - 100)) || (point.x < 100 ))

{

if ((point.y > (m\_area.height - 100)) || (point.y < 100))

{

m\_center = center;

return 0; //Нельзя изменить ни x ни y

}

m\_center.x = center.x;

return 1; //нельзя ихменить x

}

if ((point.y > (m\_area.height - 100)) || (point.y < 100))

{

m\_center.y = center.y;

return 2; //нельзя изменить y

}

return 3; //можно менять все

}

Сначала проверяется находится ли центр робота в допустимых значениях по *x.* Далее запускается проверка на допустимые значения по *y.* Если оба этих условия выполнились, то робот больше не может менять значения ни по одной из осей, следовательно координаты центра не изменяются после нажатия кнопки. Если робот вышел за границы только по *x*, то у центра перестает изменяться только переменная *x*. С *y* аналогично.

**Поворот робота**

Для поворота робота и его башни используется функция *rotate():*

int32\_t MyRobot::rotate(int8\_t buttonNumber)

{

switch (buttonNumber)

{

case 1:

{

m\_angle = m\_angle - m\_angularSpeed;

break;

}

В зависимости от того, какая кнопка была нажата, к углу робота или башни прибавляется значение *m\_angularSpeed*.

**Отрисовка робота**

Для отображения робота реализованы функции *pointСalculation()* и *drow().*

Функция *pointСalculation()* представляет из себя нахождение координат точки в глобальной системе координат:

Point2i MyRobot::pointСalculation(Point2i point, float angle)

{

Mat rotateMat = (Mat\_<float>(3, 3) <<

cos(angle), -sin(angle),m\_center.x,

sin(angle), cos(angle), m\_center.y,

0 , 0 , 1 );

Mat point1Mat = (Mat\_<float>(3, 3) <<

(point.x), 0, 0,

(point.y), 0, 0,

1, 0, 0);

Mat result;

result = rotateMat \* point1Mat;

Point2i resultXY(result.at<float>(0, 0), result.at<float>(1, 0));

return resultXY;

}

Функция выполняет следующую операцию:

*X* и *Y* – это координаты повернутой точки в глобальной системе координат. Угол – это угол поворота робота. *Tx* и *ty* это расстояние от центра глобальной системы координат до центра локальной системы. Координаты *x* и *y* – координаты точки в локальной системе.

В функции *drow()* происходит определение координат точек корпуса в локальной системе координат робота. Далее найденные точки передаются в *pointСalculation()* для определения их значений при повороте. После этого вызывается функция *line*, которая соединяет выбранные точки линией.

int32\_t MyRobot::draw(Mat& outpytImage)

{

//Определяем положение точек робота относительно центра

Point2i point1((-m\_width / 2), (m\_height / 2));

point1 = pointСalculation(point1, m\_angle);

Point2i point2((m\_width / 2), (m\_height / 2));

point2 = pointСalculation(point2, m\_angle);

Point2i point3((-m\_width / 2), (-m\_height / 2));

point3 = pointСalculation(point3, m\_angle);

Point2i point4((m\_width / 2), (-m\_height / 2));

point4 = pointСalculation(point4, m\_angle);

line(outpytImage, point1, point2, black, 3, 8, 0);

line(outpytImage, point2, point4, black, 3, 8, 0);

line(outpytImage, point3, point4, black, 3, 8, 0);

line(outpytImage, point1, point3, black, 3, 8, 0);

В функцию передается изображение, на котором нужно изобразить робота. Далее идет расчет точек и их соединение линиями.

**Пример работы программы**

