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

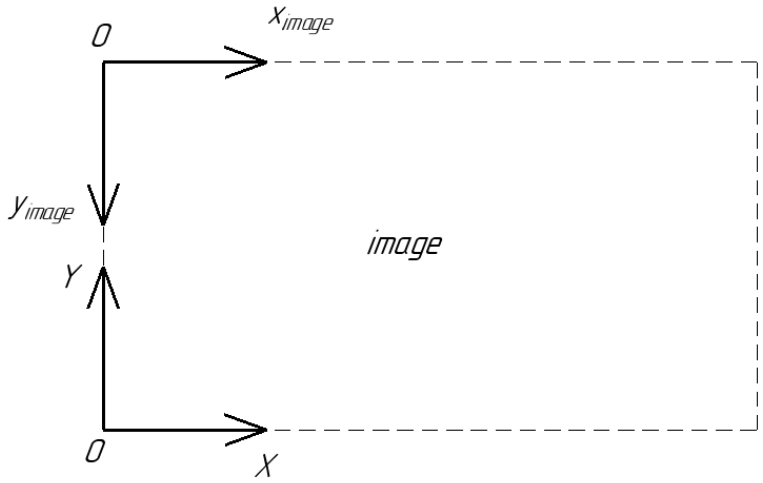
Для перемещения центра робота используются две функции: *move()* и *checkPosition()*. Первая перемещает центр в зависимости от нажатой кнопки, вторая проверяет возможно ли движение в заданном направлении.

```
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 *pointCalculation()* и *drow()*.

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

```
Point2i MyRobot::pointCalculation(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:

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

X и *Y* – это координаты повернутой точки в глобальной системе координат. Угол α – 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 *pointCalculation()* 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 = pointCalculation(point1, m_angle);
	Point2i point2((m_width / 2), (m_height / 2));
	point2 = pointCalculation(point2, m_angle);
	Point2i point3((-m_width / 2), (-m_height / 2));
	point3 = pointCalculation(point3, m_angle);
	Point2i point4((m_width / 2), (-m_height / 2));
	point4 = pointCalculation(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