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 Шкабара Я. А.

Преподаватель Варлашин В. В.

« »_____2020 г.

Санкт-Петербург

2020

Оглавление

Поставленные задачи

1. Реализовать линейное перемещение робота;
2. Реализовать поворот робота;
3. Обеспечить невозможность пересечения роботом границ поля.

Реализация линейных перемещений

Линейное перемещение выполняется при нажатии клавиши W, S, D или A. Для этого вызывается функция *moving*, принимающая направление движения в качестве аргумента:

```
int MyRobot::moving(char direction)
{
		setSpeed(direction);
		char axis = 'x';
		if (direction == 'w' || direction == 's') axis = 'y';
		int i = 0;

		float currentSpeed = getSpeed(axis);
		while (i < abs(currentSpeed))
		{
			move(direction);
			i++;
		}

	return 0;
}
```

Сначала с помощью функции *setSpeed* изменяется текущая скорость. Затем определяется текущая скорость при помощи функции *getSpeed*. Далее в цикле вызывается функция *move*, которая может переместить центр робота максимум на один пиксель за вызов: это дает возможность роботу достигать края рабочего участка. Листинг функции *move:*

```
int MyRobot::move(char direction)
{
	setAngularVelocity('c');

	float displacement_x=0, displacement_y=0;
	switch (direction)
	{
	case 'w':
		displacement_x =  sin(m_angle * PI / 180);
		displacement_y = -cos(m_angle * PI / 180);
		break;
	case 's':
		displacement_x = -sin(m_angle * PI / 180);
		displacement_y = cos(m_angle * PI / 180);
		break;
	case 'a':
		displacement_x = -cos(m_angle * PI / 180);
		displacement_y = -sin(m_angle * PI / 180);
		break;
```

```
        case 'd':
            displacement_x = cos(m_angle * PI / 180);
            displacement_y = sin(m_angle * PI / 180);
            break;
        }

        checkMovingAbility(Point2f(m_center.x + displacement_x,
    m_center.y + displacement_y),m_angle);
        return 0;
    }
```

Сначала обнуляется значение скорости поворота при помощи функции *setAngularVelocity*. Далее на основании направления движения вычисляются возможные смещения по осям *x* и *y*. В конце вызывается функция *checkMovingAbility*, проверяющая возможность движения при вычисленных смещениях.

Реализация поворота

Поворот выполняется при нажатии клавиши Q или E. Для этого вызывается функция *rotating*, принимающая направление поворота в качестве аргумента:

```
int MyRobot::rotating(char direction)
{
	setAngularVelocity(direction);
	int i = 0;

	float currentAngularVelocity = getAngularVelocity();
	while (i < abs(currentAngularVelocity))
	{
		rotate(direction);
		i++;
	}
	return 0;
}
```

Сначала с помощью функции *setAngularVelocity* изменяется текущая скорость поворота. Затем определяется текущая скорость поворота при помощи функции *getAngularVelocity*. Далее в цикле вызывается функция *rotate*, которая может повернуть робота на один градус за вызов. Листинг функции *rotate*:

```
int MyRobot::rotate(char direction)
{
	setSpeed('c');
	if (abs(m_angle) > 359) m_angle = m_angle % 360;

	switch (direction)
	{
	case 'q':
		checkMovingAbility(m_center,(m_angle-1));
		break;
	case 'e':
		checkMovingAbility(m_center,(m_angle + 1));
		break;
	}

	return 0;
}
```

Сначала обнуляется значение скорости при помощи функции *setSpeed*. Далее вызывается функция *checkMovingAbility*, проверяющая возможность поворота в указанном направлении.

Реализация проверки возможности перемещения/поворота

Проверка возможности перемещения/поворота осуществляется функцией *checkMovingAbility*, принимающей в качестве аргументов потенциально возможные положение и угол поворота, рассчитанные в функциях *rotate* и *move*. Листинг функции *checkMovingAbility:*

```
int MyRobot::checkMovingAbility(Point2f possibleCenter, int
possibleAngle)
{
	RotatedRect body(possibleCenter, Point2f(m_width, m_height),
possibleAngle);
	Point2f bodyVetrices[4];
	body.points(bodyVetrices);
	for (int i = 0; i < 4; i++)
	{
		if ((bodyVetrices[i].x > (m_area.width-1)) ||
(bodyVetrices[i].x < 0)) return -1;
		if ((bodyVetrices[i].y > (m_area.height - 1)) ||
(bodyVetrices[i].y < 0)) return -2;
	}

	RotatedRect wheels(possibleCenter, Point2f(m_width +
m_wheelWidth, m_interaxal), possibleAngle);
	Point2f wheelCenters[4];
	wheels.points(wheelCenters);
	for (int j = 0; j < 4; j++)
	{
		RotatedRect wheel(wheelCenters[j], Point2f(m_wheelWidth,
m_wheelDiameter), m_angle);
		Point2f wheelVetrices[4];
		wheel.points(wheelVetrices);
		for (int i = 0; i < 4; i++)
		{
			if ((wheelVetrices[i].x > m_area.width) ||
(wheelVetrices[i].x < 0)) return -3;
			if ((wheelVetrices[i].y > m_area.height) ||
(wheelVetrices[i].y < 0)) return -4;
		}
	}

	m_angle = possibleAngle;
	m_center = possibleCenter;
	return 0;
}
```

Сначала определяются новые координаты вершин корпуса. Затем определяется, выходят ли вершины корпуса за пределы рабочего пространства. Далее эти операции повторяются для вершин колес. Если в новом положении ни

одна точка робота не выходит за пределы рабочего пространства, то переменные, содержащие центр робота и угол поворота, меняют свои значения.

Вспомогательные функции

Функция *setSpeed* служит для обнуления и увеличения скорости движения:

```
void MyRobot::setSpeed(char direction)
{
	switch (direction)
	{
	case 'w':
		if (m_speed.y < 0) m_speed.y = 0;
		else if (m_speed.y < m_maxSpeed) m_speed.y++;
		break;
	case 's':
		if (m_speed.y > 0) m_speed.y = 0;
		else if (abs(m_speed.y) < m_maxSpeed) m_speed.y--;
		break;
	case 'a':
		if (m_speed.x > 0) m_speed.x = 0;
		else if (abs(m_speed.x) < m_maxSpeed) m_speed.x--;
		break;
	case 'd':
		if (m_speed.x < 0) m_speed.x = 0;
		else if (m_speed.x < m_maxSpeed) m_speed.x++;
		break;
	case 'c':
		m_speed.x = 0;
		m_speed.y = 0;
	}
}
```

В качестве аргумента принимает направление движения.

Функция *setAngularVelocity* служит для обнуления и увеличения скорости поворота:

```
void MyRobot::setAngularVelocity(char direction)
{
	switch (direction) {
	case 'q':
		if (m_angularVelocity > 0) m_angularVelocity = 0;
		else if (abs(m_angularVelocity) < m_maxAngularVelocity)
m_angularVelocity--;
		break;
	case 'e':
		if (m_angularVelocity < 0) m_angularVelocity = 0;
```

```
            else if (m_angularVelocity < m_maxAngularVelocity)
m_angularVelocity++;
            break;
      case 'c':
            m_angularVelocity = 0;
            break;
      }
}
```

В качестве аргумента принимает направление поворота.

Функция *stop* приравнивает линейную и угловую скорости нулю:

```
void MyRobot::stop()
{
      setAngularVelocity('c');
      setSpeed('c');
}
```

Функция *setArea* делает размеры рабочего пространства равными размерам избображения:

```
int MyRobot::setArea(Mat image)
{
      if (image.empty())
      {
            return (-1);
      }
      m_area.width = image.cols;
      m_area.height = image.rows;
      return 0;
}
```

В качестве аргумента принимает изображение.

Функция *setCenter* устанавливает координаты центра робота:

```
int MyRobot::setCenter(float x, float y)
{
      m_center.x = x;
      m_center.y = y;
      return 0;
}
```

В качестве аргумента принимает координаты *x* и *y*.

Функция *draw* отвечает за отрисовку изображения.

```
int MyRobot::draw(Mat &ioImage)
{
	if (ioImage.empty())
	{
		return -1;
	}
	if (m_area.width != ioImage.cols ||
		m_area.height != ioImage.rows)
	{
		return -2;
	}

	Mat copy = ioImage.clone();
	RotatedRect body(m_center, Point2f(m_width, m_height), m_angle);
	Point2f bodyVetrices[4];
	body.points(bodyVetrices);
	for (int i = 0; i < 4; i++) line(copy, bodyVetrices[i],
bodyVetrices[(i + 1) % 4], Scalar(0, 0, 0),2);

	RotatedRect wheels(m_center, Point2f(m_width + m_wheelWidth,
m_interaxal), m_angle);
	Point2f wheelCenters[4];
	wheels.points(wheelCenters);
	for (int j = 0; j < 4; j++)
	{
		RotatedRect wheel(wheelCenters[j], Point2f(m_wheelWidth,
m_wheelDiameter), m_angle);
		Point2f wheelVetrices[4];
		wheel.points(wheelVetrices);
		for (int i = 0; i < 4; i++) line(copy, wheelVetrices[i],
wheelVetrices[(i + 1) % 4], Scalar(0, 0, 0), 2);
	}

	imshow("It's moving!", copy);

	return 0;
	}
```

Принимает изображение в качестве аргумента и при каждом вызове создает его копию.

Функции *getSpeed* и *getAngularVelocity* возвращают линейную и угловую скорость соответственно:

```
float MyRobot::getSpeed(char direction)
{
	switch (direction)
	{
	case 'x':
		return m_speed.x;
		break;
	case 'y':
		return m_speed.y;
		break;
	}
	return 0.0f;
}

float MyRobot::getAngularVelocity()
{
	return m_angularVelocity;
}
```

getSpeed принимает ось, в направлении которой движется робот, в качестве аргумента.