САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Факультет Систем управления и робототехники

Кафедра Систем управления и информатики

Направление подготовки(специальность) 15.03.06 – Интеллектуальные технологии в робототехнике

ОТЧЕТ

о производственной практике по получению профессиональных умений и опыта профессиональной деятельности

(наименование практики)

Тема задания: Изучение ROS (Robot Operating System)

Студент Терентьев Р.А. (Фамилия И.О.) гр. Р3342 (номер группы)

Руководитель практики от организации: Капитонов А.А., кафедра СУиИ, ассистент (Фамилия И.О., должность и место работы)

Ответственный за практику от университета: Бойков В.И., доцент каф. СУиИ (Фамилия И.О., должность)

Практика пройдена с оценкой _____

Подписи членов комиссии

_____ _____
(подпись) (Фамилия И.О.)

_____ _____
(подпись) (Фамилия И.О.)

Дата _____

Санкт-Петербург
2018

Содержание

1. Цели и задачи 3

2. Написание функции распознавания контура и центра 3

3. Разбиение рабочей зоны на сектора. Определение условий движения робота для каждого сектора 5

4. Написание функции определения угла поворота для контура 6

5. Тестирование программы в симуляторе и на реальном роботе 7

6. Список используемых источников 7

7. Приложение 8

Цели и задачи

Целью данного проекта является реализация управления движением робота с помощью движений руки. Предполагается, что с помощью библиотеки OpenCV программа распознаёт руку на изображении с камеры и в зависимости от её координатных и угловых характеристик передаёт команды роботу посредством Robot Operating System. В качестве объекта управления в испытаниях используется Robotino. В ходе работы над проектом в той или иной мере были решены задачи, описанные в следующих пунктах.

Написание функции распознавания контура и центра

На этом этапе работы была поставлена задача распознавать контур и находить его центр. Программа получает на вход изображение объекта и с помощью библиотеки OpenCV выполняет поставленную задачу.

Эту часть программы можно рассмотреть, разделяя ее на несколько функций. Первая отвечает за преобразование цветового пространства.

Исходное цветное изображение нужно перевести в оттенки серого, используя функцию cvtColor. При этом цветовые каналы каждого пикселя преобразуются в один канал со значением яркости (0 представляет чёрный цвет, а 255 - белый).

Затем можно рассмотреть функцию GaussianBlur, которая отвечает за удаление лишних шумов на изображении.

Уравнение Гаусса для двух измерений выглядит так:

$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$,где x - расстояние от начала координат по горизонтальной оси, y - расстояние по вертикальной оси, а σ - стандартное отклонение гауссовского распределения. При применении в двух измерениях эта формула дает поверхность, контуры которой являются концентрическими окружностями с распределением Гаусса от центральной точки.

Далее идет функция(threshold) необходимая для разделения изображения на два класса по порогу яркости.

Пиксели с яркостью ниже пороговой окрашиваются в цвет фона (чёрный), выше пороговой – в цвет объекта (белый). Подфункция THRESH_BINARY_INV преобразует изображение в оттенках серого в двоичное изображение в соответствии с формулой:

$$dst(x,y) = \begin{cases} 0, if\ src(x,y) > threshold \\ max value, otherwise \end{cases}$$

Затем необходимо найти контуры на полученном изображении.

Контуры – это кривые, соединяющие пограничные точки одинакового цвета. Функция findContours определяет все контуры на изображении, поэтому необходимо учесть возможность попадания в область индикации других объектов. С помощью функции contourArea, которая через теорему Грина находит площадь области, ограниченной контуром, выделяем наибольший контур.

Для управления реальным объектом необходимо определить центр наибольшего контура, найденного на изображении.

Используя функцию moments, находим моменты. Момент изображения — это суммарная характеристика пятна, представляющая собой сумму всех точек (пикселей) этого пятна. При этом, имеется множество подвидов моментов, характеризующие разные свойства изображения, а затем с помощью формулы находим центр.

Чтобы получить координаты X и Y искомого пятна, нам следует поделить полученные моменты m10 и m01 на нулевой момент m00. Таким образом мы найдем средние координаты X и Y всех точек, а это и есть центр контура.

```
cap=cv2.VideoCapture(0)
_,feed=cap.read()
image=feed[0:480,0:640]
image=cv2.flip(image,1)
img=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
```

```
blur=cv2.GaussianBlur(img,(35,35),0)
h=cv2.getTrackbarPos('h','HSV_TrackBar')
ret,thresh = cv2.threshold(blur,h,255,cv2.THRESH_BINARY_INV)
_,contours,hierarchy = cv2.findContours(thresh,1,1)
max_area=0
pos=0
for i in contours:
        area=cv2.contourArea(i)
        if area>max_area:
                max_area=area
                pos=i
cnt = contours[0]
M = cv2.moments(cnt)
cx = int(M['m10']/M['m00'])
cy = int(M['m01']/M['m00'])
```

Разбиение рабочей зоны на сектора. Определение условий движения робота для каждого сектора

Мы можем видеть поле зрения камеры как прямоугольник 480*640 пикселей. Удобно разделить его на 9 зон, и при перемещении центра контура в какую-либо зону будет выполняться условие, заданное для этой зоны. В качестве условий задаются изменения скорости по двум координатам, таким образом можно изменять направление движения робота. Также коэффициенты g и v, значения которых увеличиваются по мере отдаления от центра рабочей зоны, позволяют управлять ещё и величиной скорости.

```
msg = Twist()
g,v = 1,1
if abs(cx-320)>150:
g=(abs(cx-320)/100.0)
if abs(cy-240)>130:
v=(abs(cy-240)/80.0)
…
if (cx<220 and cy<110) or (cx<170 and 110<cy<160):
msg.linear.x = 0.1*v;            msg.linear.y = 0.1*g;
elif (cx<220 and cy>370) or (cx<170 and 320<cy<370):
```

```
msg.linear.x = -0.1*v;          msg.linear.y = 0.1*g;
elif (cx>420 and cy<110) or (cx>470 and 110<cy<160):
msg.linear.x = 0.1*v;           msg.linear.y = -0.1*g;
elif (cx>420 and cy>370) or (cx>470 and 320<cy<370):
msg.linear.x = -0.1*v;          msg.linear.y = -0.1*g;
elif 220<cx<420 and cy<110:
msg.linear.x = 0.1*v;           msg.linear.y = 0;
elif 220<cx<420 and cy>370:
msg.linear.x = -0.1*v;          msg.linear.y = 0;
elif cx<170 and 160<cy<320:
msg.linear.x = 0;               msg.linear.y = 0.1*g;
elif cx>470 and 160<cy<320:
msg.linear.x = 0;               msg.linear.y = -0.1*g;
else:
msg.linear.x = 0;               msg.linear.y = 0;
```

Написание функции определения угла поворота для контура

Для возможности задавать роботу не только направление движения и скорость, но еще и контролировать с помощью контура угол поворота используется описывающий контур прямоугольник, у которого можно измерить степень наклона. Следует заметить, что повороты возможны не только во время движения, но и когда робот находиться в одной точке.

```
rect=cv2.minAreaRect(cnt)
box=cv2.boxPoints(rect)
box=np.int0(box)
x1=(box[2,0]+box[1,0])/2.0
y1=box[1,1]
x2=(box[3,0]+box[0,0])/2.0
y2=box[0,1]
ang=degrees(atan2(y2-y1,x2-x1))
…
if 15<ang<40:
msg.angular.z = -1;
elif 50<ang<75:
msg.angular.z = 1;
```

```
else:
msg.angular.z = 0;
```

Тестирование программы в симуляторе и на реальном роботе

```
pub = rospy.Publisher('cmd_vel', Twist, queue_size=10)
…
pub.publish(msg)
```

Программа связывается с роботом при помощи пакетов ROS. Данные публикуются в топик cmd_vel, который управляет колёсами робота. Примеры тестирования в симуляторе Gazebo приведены в Приложении. В ходе испытаний с Robotino удалось проверить и отладить все функции программы. Единственной проблемой, которую не удалось решить, стало неверное распознавание контуров: если в поле зрения камеры помимо руки попадал другой тёмный объект, программа могла ориентироваться на него и, следовательно, подавать роботу неверные команды.

Список используемых источников

1. ROS. Официальный сайт. [Электронный ресурс]/ Режим доступа: www.ros.org
2. RobotinoWiki [Электронный ресурс]/ Режим доступа: wiki.openrobotino.org
3. OpenCV: OpenCV tutorials [Электронный ресурс]/ Режим доступа: docs.opencv.org

Приложение

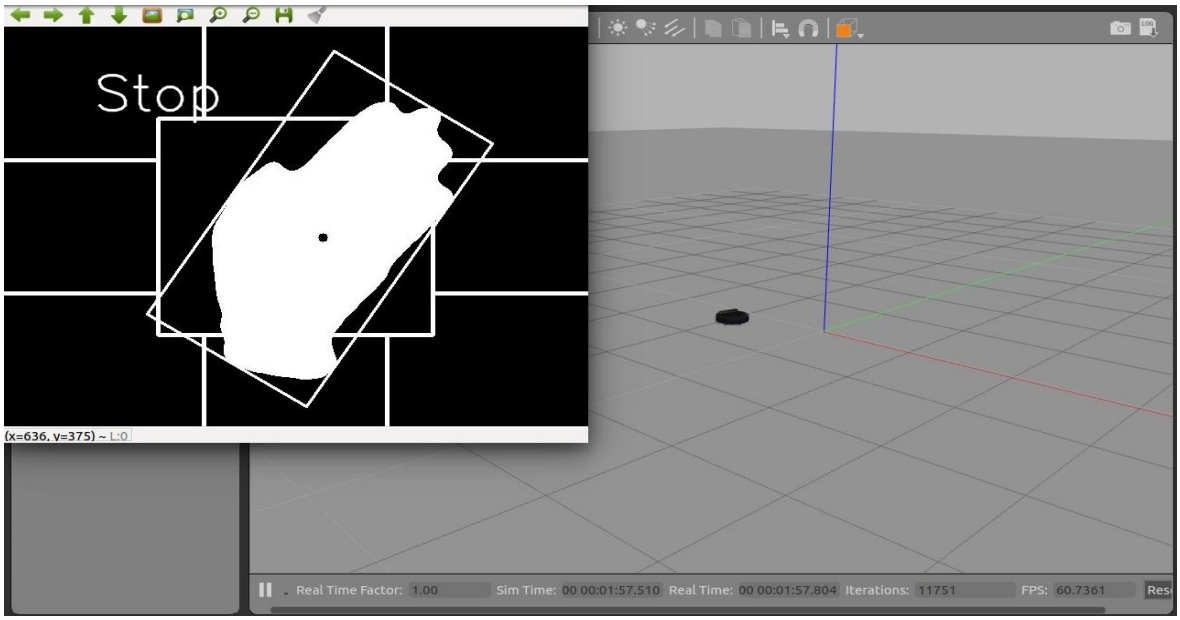

Рисунок 1. Робот вращается на месте

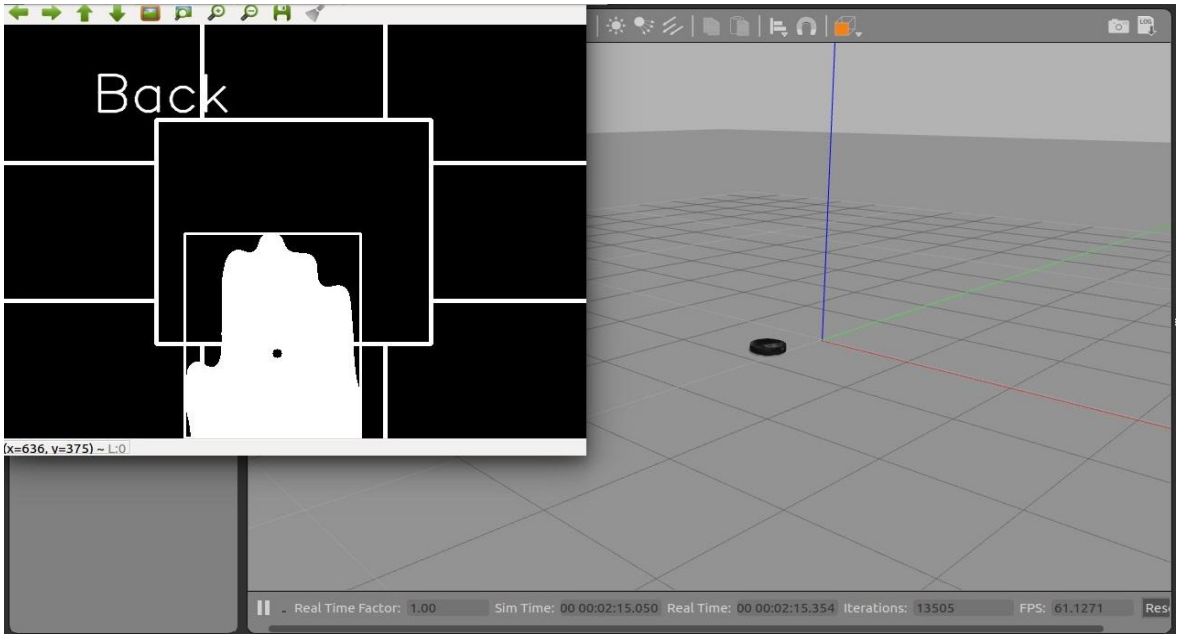

Рисунок 2. Робот движется назад с минимальной скоростью