УДК 004.932

РАЗРАБОТКА СИСТЕМЫ РАСПОЗНАВАНИЯ ОБРАЗОВ ДЛЯ МОБИЛЬНОГО РОБОТА

Ульянов Сергей Викторович[1], Решетников Андрей Геннадьевич[2], Кошелев Кирилл Викторович[3]

[1]Доктор физико-математических наук, профессор;
ГБОУ ВО МО «Университет «Дубна»,
Институт системного анализа и управления;
141980, Московская обл., г. Дубна, ул. Университетская, 19;
e-mail: ulyanovsv@mail.ru.

[2]Доктор информатики, ассистент;
ГБОУ ВО МО «Университет «Дубна»,
Институт системного анализа и управления;
141980, Московская обл., г. Дубна, ул. Университетская, 19;
e-mail: agreshetnikov@gmail.com.

[3]Студент;
ГБОУ ВО МО «Университет «Дубна»,
Институт системного анализа и управления;
141980, Московская обл., г. Дубна, ул. Университетская, 19;
e-mail: kirill_koshelev18@rambler.ru.

В данной статье приводится краткое описание разрабатываемой системы распознавания образов. Описывается используемая в работе технология стереозрения, рассматриваются ее преимущества. В работе представлено описание настройки используемого оборудования. В качестве результата представлен программный модуль распознавания, разработанный с помощью языка программирования Python.

Ключевые слова: распознавание образов, стереозрение, *Robot Operating System (ROS)*.

DEVELOPMENT OF PATTERN RECOGNITION SYSTEM FOR A MOBILE ROBOT

Ulyanov Sergey[1], Reshetnikov Andrey[2], Koshelev Kirill[3]

[1]Doctor of Science in Physics and Mathematics, professor;
Dubna State University,
Institute of the system analysis and management;
141980, Dubna, Moscow reg., Universitetskaya str., 19;
e-mail: ulyanovsv@mail.ru.

[2]PhD, assistant professor;
Dubna State University,
Institute of the system analysis and management;
141980, Dubna, Moscow reg., Universitetskaya str., 19;
e-mail: agreshetnikov@gmail.com.

[3]Student;
Dubna State University,
Institute of the system analysis and management;
141980, Dubna, Moscow reg., Universitetskaya str., 19;
e-mail: kirill_koshelev18@rambler.ru.

This article provides a brief description of the developed pattern recognition system. Describes stereovision technology, discusses its advantages. The paper presents a description of the equipment settings. Result of the work is the recognition module developed with the Python programming language.

Keywords: stereovision, pattern recognition, Robot Operating System (ROS).

Введение

Основным источником развития робототехнических систем и систем искусственного интеллекта, является развитие вычислительной техники, устройств, и сенсоров, позволяющих получать информацию об окружающей среде. Данный процесс несомненно связанно с развитием программного обеспечения бортовых процессоров, для управления автономным роботом. В настоящее время, все чаще упоминаются системы, состоящие из комбинаций нескольких видео камер, лазерных радаров, тепловизоров и других источников получения информации об окружающей среде. Широкое применение оборудование такого класса получило при создании автопилотов для транспортных средств, систем контроля безопасности, беспилотной авиаразведки, различных бытовых роботов [2, 7] и т.д.

Разрабатываемая система распознавания базируется на технологии стереозрения. Применение стереозрения позволяет получить данные о глубине изображения, расстоянии до объектов, предоставляет возможность строить трехмерную картину окружающего мира. Соответственно, с одной стороны, разработчик сталкивается с изобилием информационных источников, с другой стороны, возникает сложность эффективного комбинирования и построения системы управления для автономного робота. Соответственно, процедура проектирования и настройки системы управления сводится к настройке множества параметров управления, что в свою очередь вызывает трудности у экспертов.

В технологии стереозрения используются, как правило, две камеры, работающие синхронно, что позволяет восстанавливать форму и расположение наблюдаемых объектов в трехмерном пространстве. При этом создается трехмерная полигональная модель какого-либо объекта, формируемая на основе анализа изображений этого объекта, полученных разнесенными в пространстве камерами.

Обычно для машинного зрения и распознавания достаточно использовать одну камеру, существует множество примеров работ [3, 8, 12], но стоит отметить, что стереозрение, в определенной степени повторяя особенности развития природного зрения, позволяет бортовой системе получать информацию не только о цвете и яркости объекта, но и о расстоянии до него, о его геометрической форме, о препятствия на пути к объекту. Помимо этого, совместное использование различных сенсоров и их комбинаций с технологией стереозрения позволяет более качественно и полно строить «сцену мира» для робота, улучшая тем самым его взаимодействие с окружающей средой. К таким сенсорам относятся сонары и лидары, радары, их применение, естественно, повышает себестоимость технического решения и сложность программного интеллекта бортовой системы.

Тематика публикации выбрана не случайно, в рамках работы над магистерской диссертацией “Разработка системы распознавания образов на основе интеллектуальных вычислений” был разработан программный модуль распознавания объектов для операционной системы робота *Robot Operating System* (далее *ROS*) [6]. Выполнение данной задачи диссертации состояло из следующих этапов:

1. настройка стереопары из двух камер;
2. построение карты смещений *(disparity map)*;
3. использование карты смещений для получения облака точек при помощью 3D-визуализатора *Rviz* (*ROS visualization*);
4. создание программного модуля распознавания объектов на языке программирования *Python*.

Соответственно, результатом выполнения данной задачи стало реализованное программное обеспечения для фреймворка *ROS*, которое в дальнейшем станет базисом интеграции интеллектуальных вычислений.

1. Настройка стереопары из двух камер

Настройка стереопары из двух камер является первым этапом при проектировании системы стереозрения в *ROS*. Для выполнения поставленной задачи была использована пара камер, изображенная на рисунке 1.

Рис. 1. Используемые камеры

Для обеспечения возможности обмена данными между *ROS* и камерами необходимо создать в рабочем каталоге, который связан с *ROS*, пакет с файлом инициализации. В файле содержатся параметры для работы и изображением, а именно разрешение, имена видеоустройств (рис. 2).

rec.py × calibr.py × dr.launch ×

```
<launch>
        <node name="left" pkg="usb_cam" type="usb_cam_node" output="screen" >
    <param name="video_device" value="/dev/video1"></param>
    <param name="image_width" value="640"></param>
    <param name="image_height" value="480"></param>
    <param name="pixel_format" value="yuyv"></param>
    <param name="camera_frame_id" value="usb_cam"></param>
    <param name="io_method" value="mmap"></param>
  </node>
  <node name="right" pkg="usb_cam" type="usb_cam_node" output="screen" >
    <param name="video_device" value="/dev/video2"></param>
    <param name="image_width" value="640"></param>
    <param name="image_height" value="480"></param>
    <param name="pixel_format" value="yuyv"></param>
    <param name="camera_frame_id" value="usb_cam"></param>
    <param name="io_method" value="mmap"></param>
  </node>
<node name="image_view" pkg="image_view" type="image_view" respawn="false" output="screen">
    <remap from="image" to="/stereo/left/image_raw"/>
    <param name="autosize" value="true"></param>
  </node>
<node name="imager_view" pkg="image_view" type="image_view" respawn="false" output="screen">
    <remap from="image" to="/stereo/right/image_raw"/>
    <param name="autosize" value="true"></param>
  </node>
</launch>
```

Рис. 2. Файл запуска

В тексте файла видно, что каждая камера, по сути, становится узлом с определенным именем, а узлы, в свою очередь, взаимодействуют с *ROS* [6]. Выполнив команды, представленные на рис. 3 для проверки правильности выполненных действий, получим изображение с используемых камер (рис.4),

```
/home/robot/catkin_ws/src/telega_vehicle/launch/dr.launch http://localhost:11311
robot@robot-W65-67SF:~$ ROS_NAMESPACE=stereo roslaunch telega_vehicle dr.launch
WARNING: Package name "tinyIMU_relay" does not follow the naming conventions. It
 should start with a lower case letter and only contain lower case letters, digi
ts and underscores.
... logging to /home/robot/.ros/log/d0707dc2-74c2-11e6-b5ac-acb57d1cd88c/roslaun
ch-robot-W65-67SF-7947.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
WARNING: disk usage in log directory [/home/robot/.ros/log] is over 1GB.
It's recommended that you use the 'rosclean' command.

started roslaunch server http://robot-W65-67SF:37075/

SUMMARY
========

PARAMETERS
 * /rosdistro: jade
 * /rosversion: 1.11.16
 * /stereo/image_view/autosize: True
 * /stereo/imager_view/autosize: True
 * /stereo/left/camera_frame_id: usb_cam
 * /stereo/left/image_height: 480
 * /stereo/left/image_width: 640
```

Рис. 3. Запуск камер

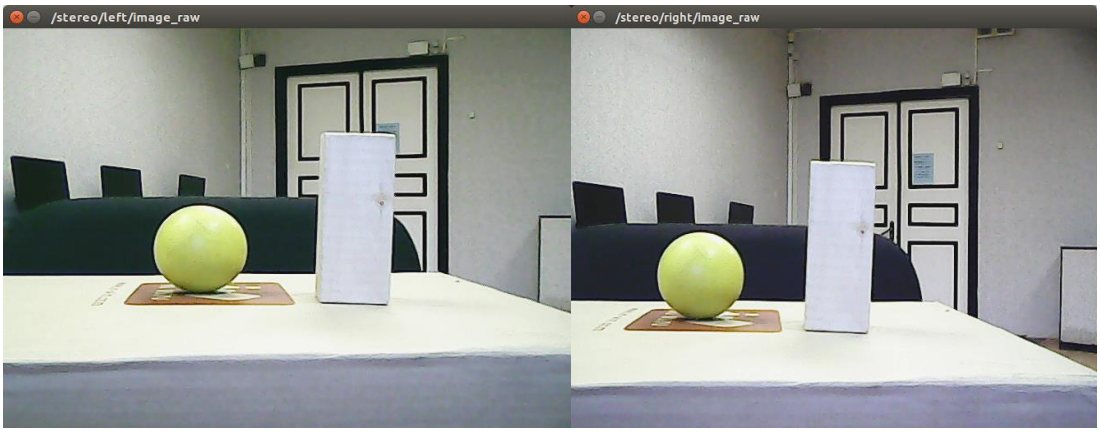

Рис. 4. Изображения, полученные с камер

После успешного запуска камер можно переходить к калибровке. Калибровка камер позволяет исправлять дисторсию (аберрацию оптических систем), при которой коэффициент линейного увеличения изменяется по полю зрения объектива. При этом нарушается геометрическое подобие между объектом и его изображением. Калибровка стереопары будет производиться с помощью фреймворка *ROS* методом шахматной доски.

Калибровка камер обычно выполняется за счет многократной съемки калибровочного шаблона, на изображении можно легко выделить ключевые точки (рис. 5), для которых известны их относительные положения в пространстве. Затем находятся коэффициенты, связывающие координаты проекций, матрицы камер и положения точек шаблона в пространстве [4, 6].

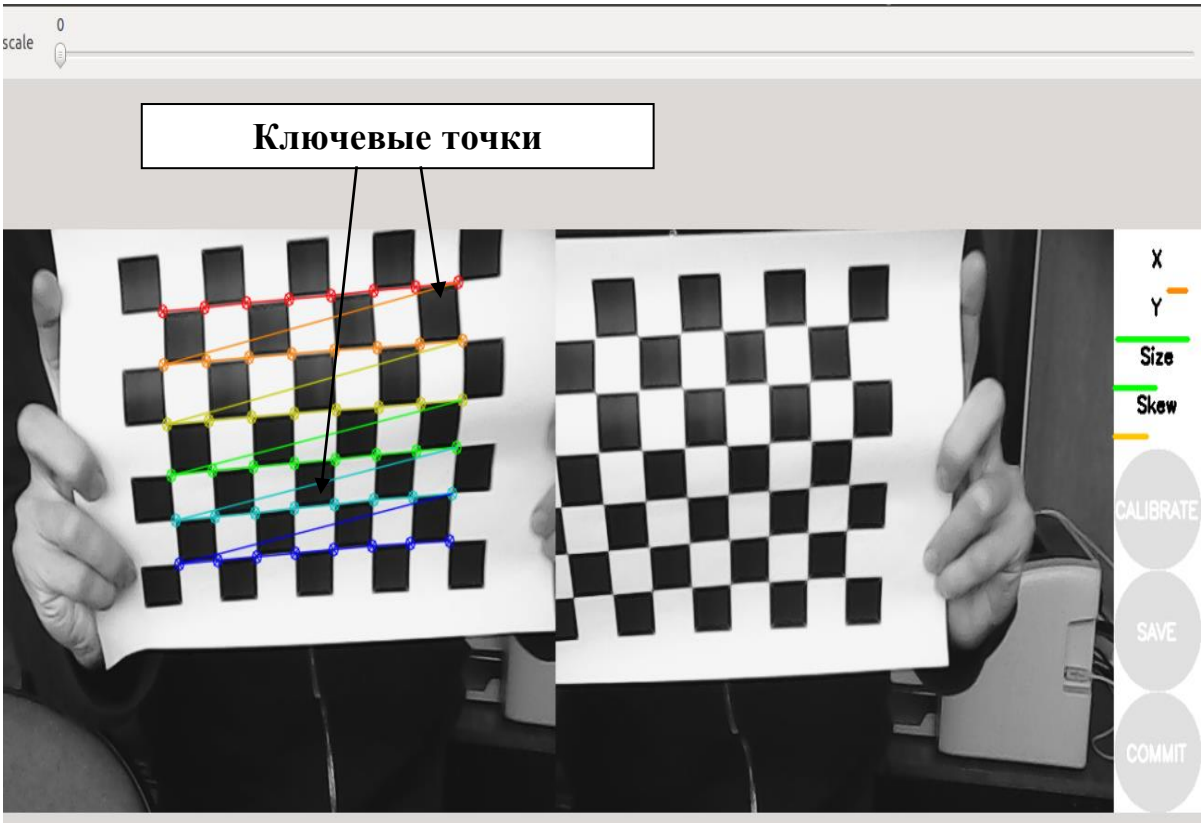

Рис. 5. Процесс калибровки (выделение ключевых точек)

2. Построение карты смещений (disparity map)

Для построения трехмерного изображения с помощью 3*D*-визуализатора *Rviz (ROS visualization)* используются карты глубины и карты смещений.

Для начала рассмотрим такое понятие как карта глубины [5]. Карта глубины *(depth map)* — это способ представления объемных 3*D* изображений в виде изображения, в котором каждому пикселю присваивается дополнительный параметр – глубина. Этот параметр показывает, на каком расстоянии от плоскости изображения расположен данный пиксель. Карта глубины может быть получена с помощью специальной камеры глубины (например, сенсор *Kinect* является своего рода такой камерой), а так же может быть построена по изображениям, полученным со стереопары (подобные камеры начинают применяться, в том числе, и в мобильных устройствах).

При построении карты смещений для каждой точки на одном изображении выполняется поиск парной ей точки на другом изображении. По паре соответствующих точек можно выполнить триангуляцию и определить координаты их прообраза в трехмерном пространстве. При известных трехмерных координатах прообраза глубина вычисляется как расстояние до плоскости камеры (рис. 6).

Для каждой точки на изображении с первой камеры соответствующую ей парную точку нужно искать в той же строчке на изображении со второй камеры. Для каждого пикселя левого изображения с координатами (x_0, y_0) выполняется поиск соответствующего пикселя на правом изображении. При этом предполагается, что пиксель на правом изображении должен иметь координаты $(x_0 - d, y_0)$, где d – величина, называемая смещение (*disparity*) (рис. 6) [1, 11]. Поиск соответствующего пикселя выполняется путем вычисления максимума функции отклика, в качестве которой может выступать, например, корреляция окрестностей пикселей. В результате получается карта смещений *(disparity map)*.

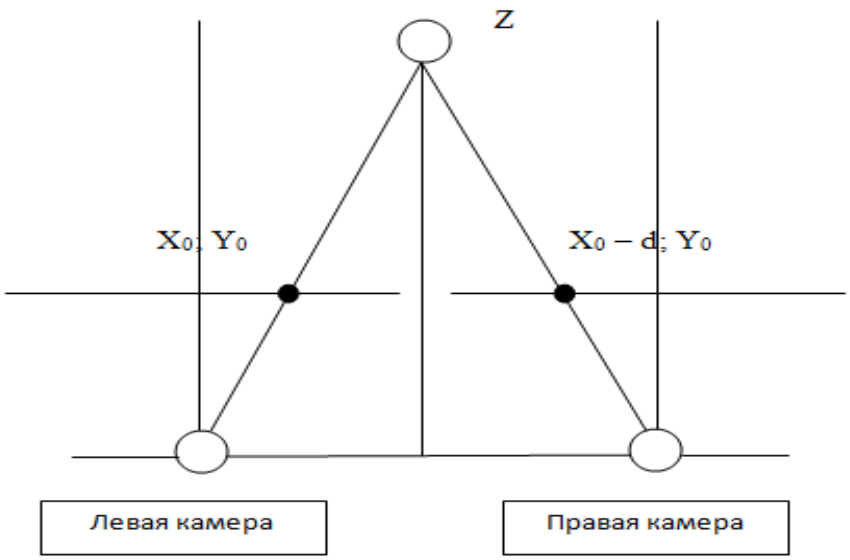

Рис. 6. Поиск соответствующих пар точек

В результате получается карта смещений *(disparity map)*, пример которой приведен на рисунке 7.

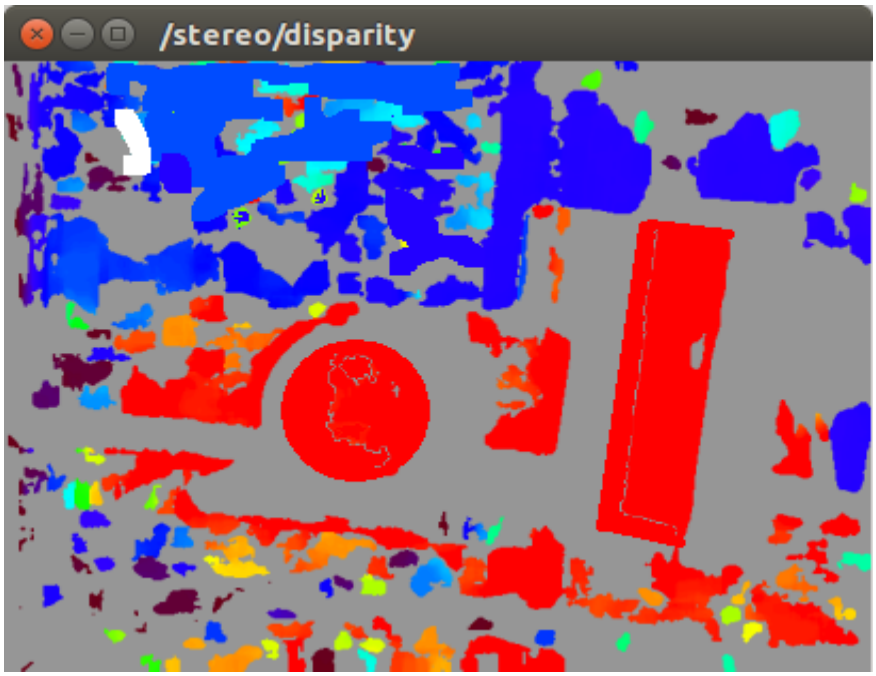

Рис. 7. Полученная карта смещений

Карта строится на основе изображений, которые мы получили ранее (рис. 4).

3. Получение облака точек с помощью 3D-визуализатора Rviz (ROS visualization)

3*D*-визуализатор *Rviz* – модуль фреймворка *ROS*, предназначенный для отображения данных с различных датчиков (включая камеры, сонары, дальномеры). С его помощью, например, можно построить трехмерную карту помещения, а робот под управлением *ROS*, используя полученные данные об объектах и препятствиях, сможет автономно перемещаться в этом помещении. Первоначально в области глобальных настроек *Global Options* напротив поля *Fixed Frame* указываем

имя пакета, посредством которого *ROS* взаимодействует с камерами *(usb_cam)*. Затем нажимаем кнопку *Add* и выбираем из списка *PointCloud2* (рис. 8).

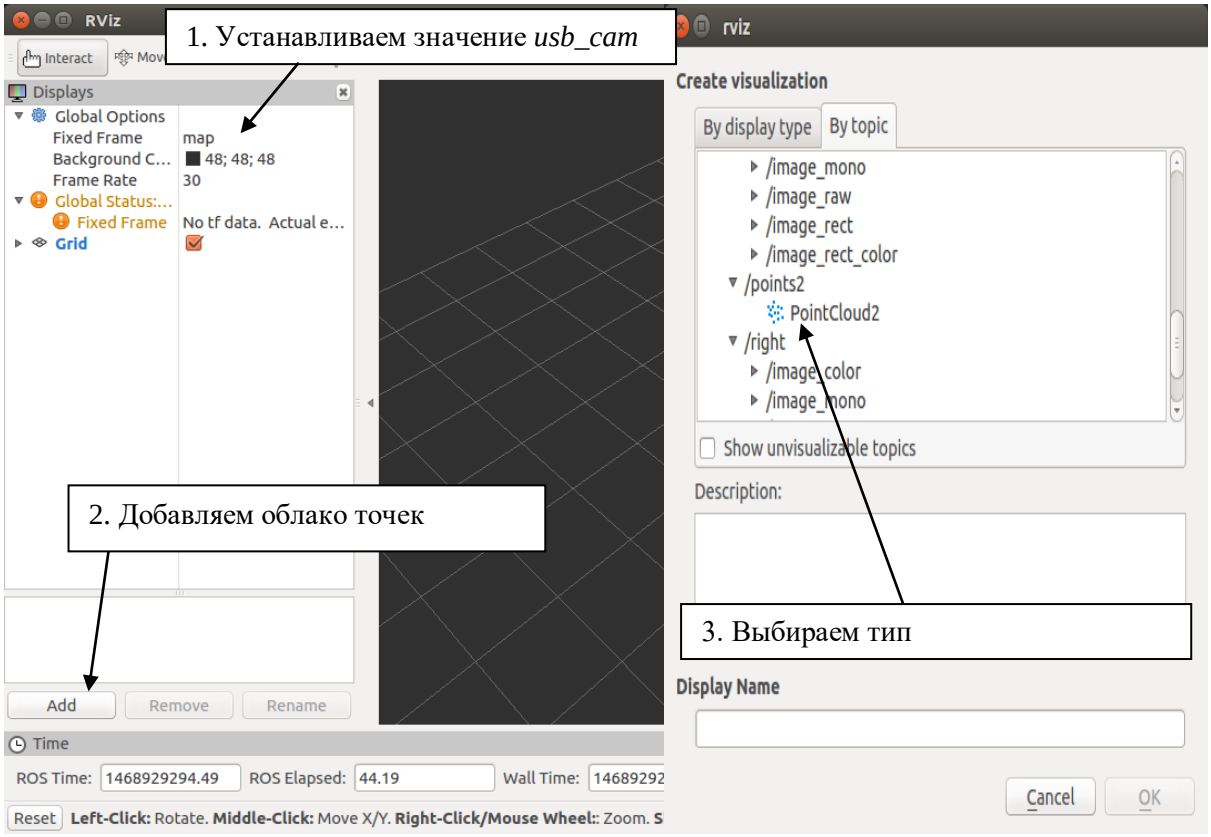

Рис. 8. Добавление облака точек

Сообщение *sensor_msgs/PointCloud2* описывает облако точек в трехмерном пространстве [6]. На рис. 9 показано трехмерное изображение, полученное с помощью стереопары из двух камер.

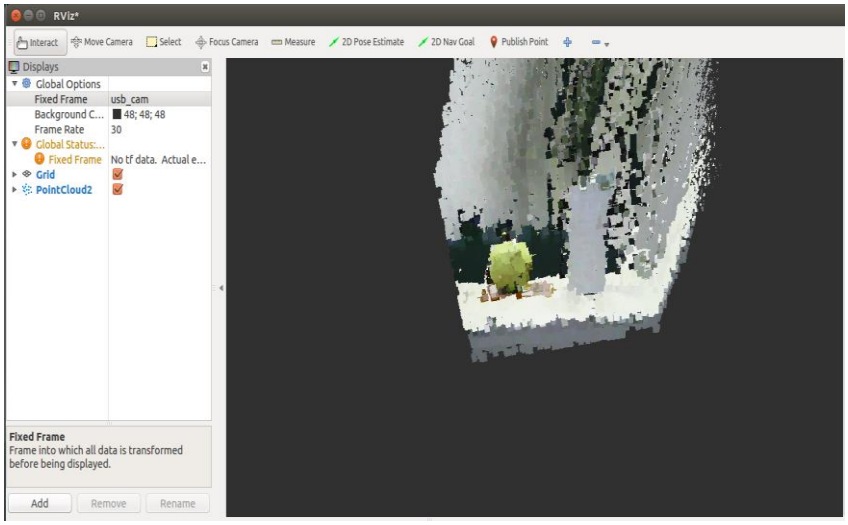

Рис. 9. 3D-изображение, полученное с помощью визуализатора Rviz

4. Программный модуль системы распознавания

При выполнении задачи по распознаванию объектов программный код создавался на языке программирования *Python*. Получение и обработка изображений фреймворком *ROS* происходили при

помощи настроенных ранее камер. В качестве объекта был выбран шар желтого цвета. Разработанный модуль производит распознавание и слежение за объектом (рис. 10).

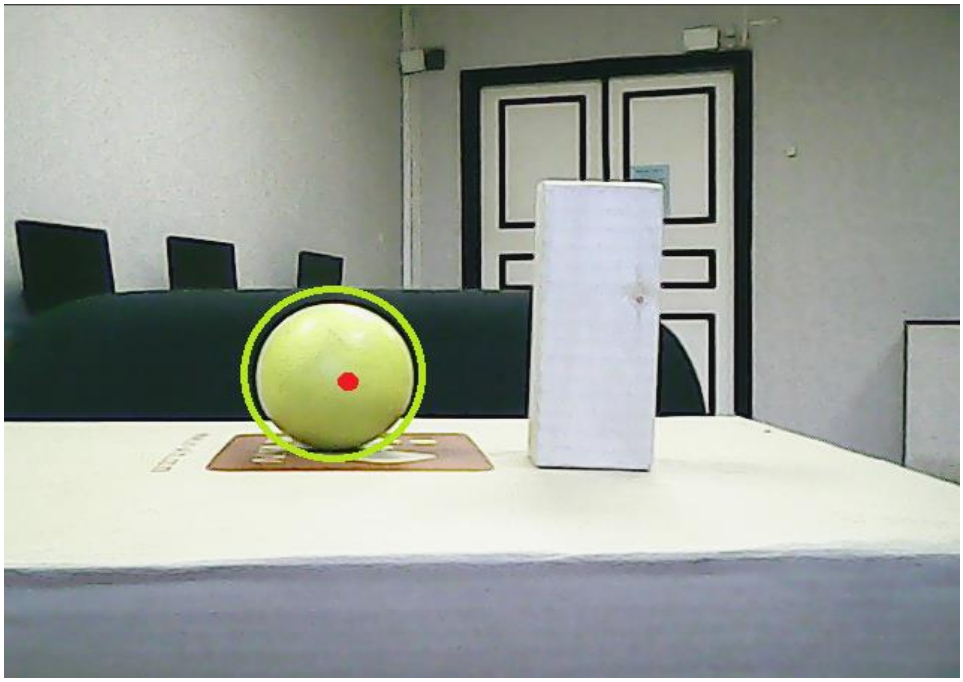

Рис. 10. Распознавание шара

На рисунке 10 видно, что программа выделяет контуры объекта. Пример программного кода приведен на рисунке 11.

rec.py × calibr.py × dr.launch × cam.launch ×

```
def callbackR(self,data):
        try:
                image = self.bridge.imgmsg_to_cv2(data, "bgr8")
        except CvBridgeError, e:
                print e

        h = cv2.getTrackbarPos('dp','mask')
        s = cv2.getTrackbarPos('mindist','mask')
        v = cv2.getTrackbarPos('p1','mask')
        h1 = cv2.getTrackbarPos('p2','mask')
        s1 = cv2.getTrackbarPos('minR','mask')
        v1 = cv2.getTrackbarPos('maxR','mask')

        greenLower = (22, 34, 172)
        greenUpper = (56, 255, 255)
        pts = deque(maxlen=64)

        image = imutils.resize(image, width=600)
        hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

        # construct a mask for the color "green", then perform
        # a series of dilations and erosions to remove any small
        # blobs left in the mask
        mask = cv2.inRange(hsv, greenLower, greenUpper)
        mask = cv2.erode(mask, None, iterations=2)
        mask = cv2.dilate(mask, None, iterations=2)
        cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
                cv2.CHAIN_APPROX_SIMPLE)[-2]
        center = None


        gray=mask
        st1=cv2.getStructuringElement(cv2.MORPH_RECT,(21,21),(10,10))
        st2=cv2.getStructuringElement(cv2.MORPH_RECT,(11,11),(5,5))
        gray = cv2.GaussianBlur(gray,(5,5),2)
```

Рис. 11. Программный код

Стоит также отметить, что данный модуль позволяет осуществлять слежение за объектом. Однако программа распознает лишь те объекты, форма и цвет которых указаны непосредственно в коде, что несколько снижает ее эффективность (рис. 12).

rec.py × calibr.py × dr.launch × cam.launch ×

```
gray=cv2.morphologyEx(gray,cv2.MORPH_CLOSE,st1)
gray=cv2.morphologyEx(gray,cv2.MORPH_OPEN,st1)
circles = cv2.HoughCircles(gray.copy(),cv.CV_HOUGH_GRADIENT,h+1,500,
param1=v+1,param2=h1+1,minRadius=s1,maxRadius=v1)

try:
        l = len(circles[0,:])
        for i in circles[0,:]:
                cv2.circle(image,(i[0],i[1]),i[2],(0,0,255),5)
                center2=(i[0],i[1])
                r=i[2]
except TypeError:  pass

if len(cnts) > 0:
        # find the largest contour in the mask, then use
        # it to compute the minimum enclosing circle and
        # centroid
        c = max(cnts, key=cv2.contourArea)
        ((x, y), radius) = cv2.minEnclosingCircle(c)
        M = cv2.moments(c)
        center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))

        # only proceed if the radius meets a minimum size
        if radius > 10 :
                # draw the circle and centroid on the frame,
                # then update the list of tracked points
                cv2.circle(image, (int(x), int(y)), int(radius),
                        (0, 255, 255), 2)
                cv2.circle(image, center, 5, (0, 0, 255), -1)

# update the points queue
pts.appendleft(center)
for i in xrange(1, len(pts)):
        # if either of the tracked points are None, ignore
        # them
        if pts[i - 1] is None or pts[i] is None:
```

Рис. 12. Программный код (продолжение)

Решить данную проблему позволит введение качества самоорганизации в процесс распознавания. Подобные вопросы уже рассматривались в научно-исследовательской группе университета «Дубна».

Настоящая работа носит вводный характер для рассмотрения вопросов внедрения программного инструментария на основе мягких и квантовых вычислений в процесс проектирования системы распознавания и системы управления автономного мобильного робота.

Для повышения точности и качества распознавания при разработке системы будут использоваться интеллектуальные вычисления (использование нейросетевого подхода с элементами нечеткой логики и эволюционных алгоритмов обучения) [9, 10].

Заключение

В рамках данной статьи рассмотрена технология стереозрения, которая в настоящее время все более активно применяется в робототехнике. Правильная настройка стереокамер позволяет значительно повысить качество распознавания объектов. Использование двух камер (стереопары) дает возможность подсчитывать расстояние до объектов, а также получать трехмерное изображение окружающего пространства, что значительно упростит управление мобильным роботом, в который будет интегрирован данный модуль распознавания.

Изобилие систем машинного зрения не устраняет главные недостатки систем распознавания (погрешность распознавания при изменении ракурса объекта, изменение освещения, чувствительность ПО и т.д.) [10]. Использование интеллектуальных вычислений в совокупности с технологией стереозрения позволит создать эффективную систему распознавания, которая будет лишена (полностью или частично) перечисленных выше недостатков.

Список литературы

1. Bobick A. F., Intille S. S. Large occlusion stereo // Int. Journal of Computer Vision – 1999. – Vol. 33 – Pp. 181-200.
2. Engineering company Neptec: instruments and equipment. – [Электронный ресурс]. URL: http://www.neptec.com/technology/.

3. Expanding object detector's Horizon: Incremental Learning Framework for object detection in videos. – [Электронный ресурс]. URL: https://www.disneyresearch.com/publication/expanding-object-detectors-horizon/.

4. Janusz Szpytko, Pawel Hyla. Stereovision 3D type workspace mapping system architecture for transport devices // Journal of KONES Powertrain and Transport – 2010. – Vol. 17 – Pp. 496-502.

5. Pentland A. P. Depth of Scene from Depth of Field // Proc. Image Understanding Workshop. – 1982. – Pp. 253-259.

6. Robot Operating System (ROS): documentation, support, tutorials. – [Электронный ресурс]. URL: http://wiki.ros.org/.

7. ИПМТ ДВО РАН: подводная робототехника. – [Электронный ресурс]. URL: http://www.imtp.febras.ru/podvodnaya-robototexnika.html?showall=1.

8. Распознавание образов для программистов: примеры работ. – [Электронный ресурс]. URL: http://recog.ru/.

9. Ульянов С. В., Добрынин В. Н., Нефедов Н. Ю., Петров С. П., Полунин А. С., Решетников А. Г. Генетические и квантовые алгоритмы. Ч. 1. Инновационные модели в обучении // Системный анализ в науке и образовании: электрон. науч. журнал. – 2010. – №3. – [Электронный ресурс]. URL: http:/www.sanse.ru/archive/17. – 0421000111\0030.

10. Ульянов С. В., Петров С. П. Квантовое распознавание лиц и квантовая визуальная криптография: модели и алгоритмы // Системный анализ в науке и образовании: электрон. науч. журнал. – 2012. – №1. – [Электронный ресурс]. URL: http:/www.sanse.ru/archive/23. – 0421200111\0007.

11. Форсайт Д. А., Понс Ж. Компьютерное зрение: современный подход. – М.: Вильямс, 2004. – С. 928.

12. Яне Б. Цифровая обработка изображений. – М.: Техносфера, 2007. – С. 584.