–МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**“УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”**

Факультет информационных систем и технологий

Кафедра “ Вычислительная техника”

Дисциплина «Программирование робототехнических комплексов»

**Лабораторная работа №3**

«Распознавание объектов с использованием OpenCV»

Вариант 12

Выполнил:

студент гр. ИВТАПбд-31

Кондратьев П.С.

Проверил:

Святов К.В.

Ульяновск, 2019

**1. Техническое задание**

В среде V-REP на языке программирования python необходимо разработать программу, которая позволяет распознавать объекты определенного типа, заданного вариантом. В качестве сцены необходимо использовать сцену, разработанную в лабораторной работе №1. Вариант по заданию –синий цилиндр.

**2. Ход работы**

Для данной лабораторной работы была создана сцена, в которую были добавлены простые объекты (кубы и сферы, цилиндры) разных цветов (см рис. 1).

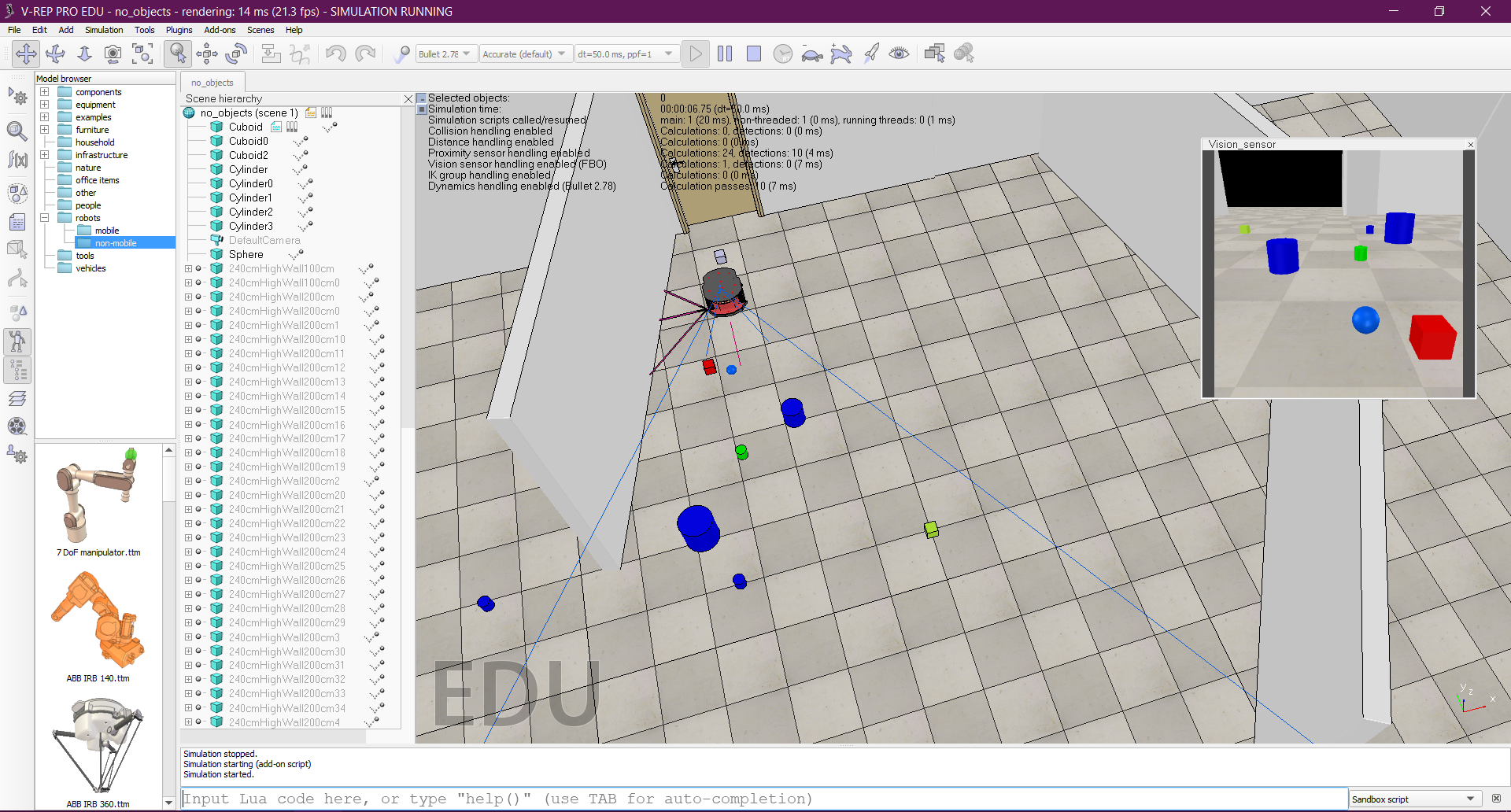


Рис. 1. Сцена для лабораторной работы.

К роботу Pioneer\_3dx необходимо прикрепить датчик зрения (vision sensor), который будет получать изображение, необходимое для передачи и дальнейшего анализа картинки с камеры в библиотеке OpenCV.

Для распознавания объектов был написан скрипт, который перво-наперво проверяет соединения, в случае успеха происходит получение объекта – датчика зрения. После чего в цикле происходит получение изображения и его первичная обработка (переворачивание и т.п.), затем задаются цветовые границы для определяемых объектов в формате hsv (для синего цвета были выбраны следующие значения: нижняя граница – 82, 25, 0; а для верхней границы – 161, 255, 255). Происходит конвертация изображения в формат. Для избавления от возможных маленьких объектов, которые могут быть похожими на цилиндр, находится площадь контура.

Если объект не признается маленьким, то для него производится контурная аппроксимация, после чего по количеству точек определяется содержит ли текущий контур объект, похожий на цилиндр, или нет.

Вся визуальная информация выводится в отдельное окно, а котором отображаются: изображение с сенсора (на изображение накладываются прямоугольники, в которых находятся синие цилиндры, если такие имеются.

На рисунках 2-4 изображены скриншоты работы программы для определения синих цилиндров (на камере они показываются красным цветом). На рисунке 4 можно увидеть пример, в котором определяется синий цилиндр (красный), но не зеленый цилиндр. На рисунке 3 – пример, в котором определяются синий цилиндр, но не дальний цилиндр, поскольку в поле зрения сенсора он не попадает.

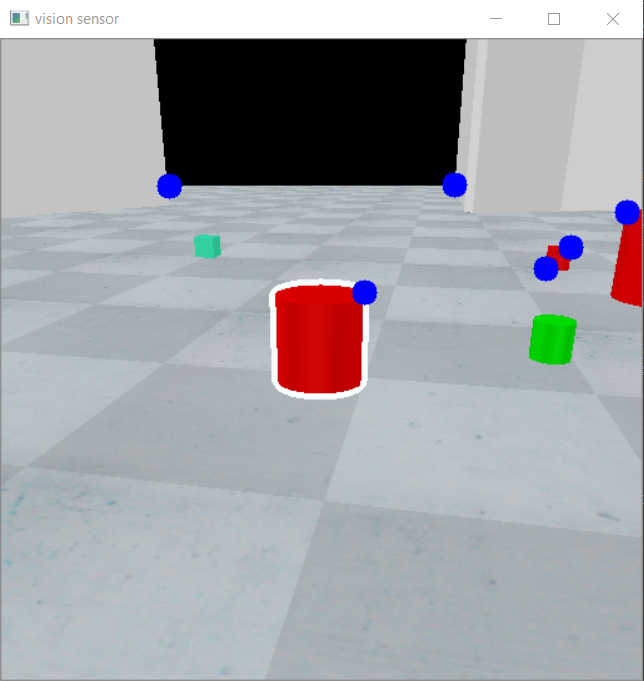


Рис. 2. Результат работы программы №1

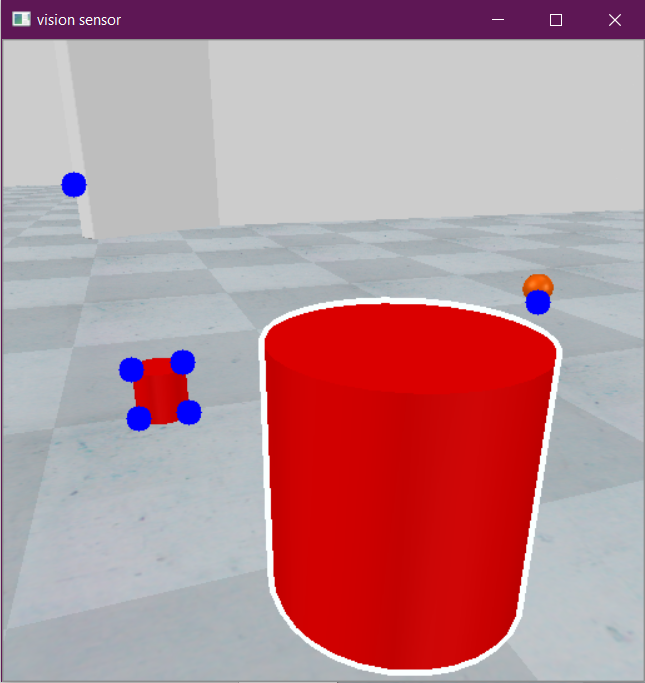


Рис. 3. Результат работы программы №2

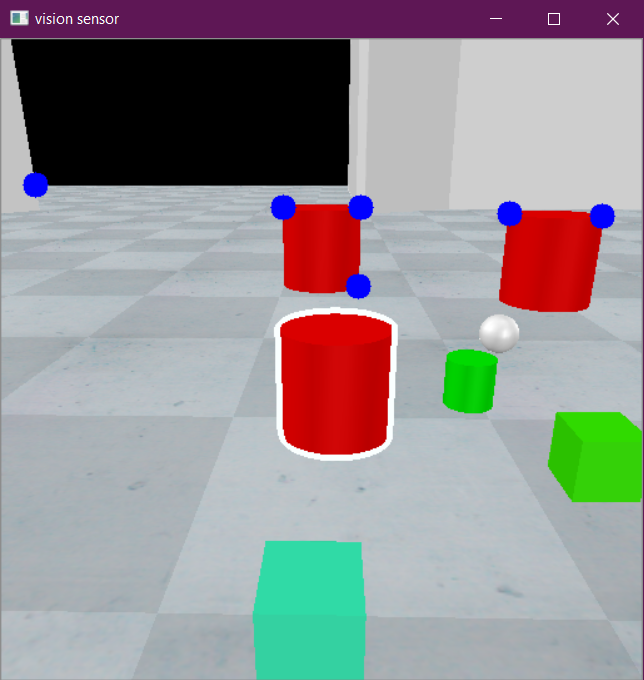


Рис. 4. Результат работы программы №3

**3. Листинг программы**

import vrep  
import sys  
import cv2  
# Importing the Opencv Library  
import numpy as np  
# Importing NumPy,which is the fundamental package for scientific computing with #Python  
vrep.simxFinish(-1)  
clientId = vrep.simxStart('127.0.0.1', 19997, True, True, 5000, 5)  
if clientId != -1:  
 print("Connected to remote server")  
else:  
 print('Connection not successful')  
 sys.exit('Could not connect')  
  
print("Press \"Start simulation in V-REP\"")  
  
while vrep.simxGetConnectionId(clientId) != -1:  
 res, visionSensor = vrep.simxGetObjectHandle(clientId, 'Vision\_sensor', vrep.simx\_opmode\_oneshot\_wait)  
 err, resolution, image = vrep.simxGetVisionSensorImage(clientId, visionSensor, 0, vrep.simx\_opmode\_streaming)  
  
 if err == vrep.simx\_return\_ok:  
 img = np.array(image, dtype=np.uint8)  
 img.resize([resolution[1], resolution[0], 3])  
 img = cv2.flip(img, 0)  
 cv2.imshow('vision sensor', img)  
 # RGB to Gray scale conversion  
 img\_gray = cv2.cvtColor(img,cv2.COLOR\_RGB2GRAY)  
 noise\_removal = cv2.bilateralFilter(img\_gray,9,75,75)  
 ret,thresh\_image = cv2.threshold(noise\_removal,0,255,cv2.THRESH\_OTSU)  
 canny\_image = cv2.Canny(thresh\_image,250,255)  
 canny\_image = cv2.convertScaleAbs(canny\_image)  
  
 # dilation to strengthen the edges  
 kernel = np.ones((3,3), np.uint8)  
 # Creating the kernel for dilation  
 dilated\_image = cv2.dilate(canny\_image,kernel,iterations=1)  
 # Displaying Image  
 contours, h = cv2.findContours(dilated\_image, 1, 2)  
 contours= sorted(contours, key = cv2.contourArea, reverse = True)[:1]  
 pt = (180, 3 \* img.shape[0] // 4)  
 for cnt in contours:  
 approx = cv2.approxPolyDP(cnt,0.01\*cv2.arcLength(cnt,True),True)  
 if len(approx) == 8:  
 cv2.drawContours(img, [cnt], -1, (255, 255, 250), 3)  
 corners = cv2.goodFeaturesToTrack(thresh\_image, 6, 0.06, 25)  
 corners = np.float32(corners)  
 for item in corners:  
 x, y = item[0]  
 cv2.circle(img, (x, y), 10, 255, -1)  
 cv2.imshow('vision sensor', img)  
  
 if cv2.waitKey(1) & 0xFF == ord('q'):  
 break  
vrep.simxFinish(clientId)  
  
cv2.waitKey()

**4. Выводы по лабораторной работе**

В ходе выполнения данной лабораторной работы удалось создать программу с использованием библиотеки OpenCV, позволяющую определять простые необходимые объекты по изображению с визуального датчика. При определении фигур проблематичным является определение типа фигуры, для чего пришлось прибегнуть к контурной аппроксимации, также в ходе выполнения работы возникали проблемы с определением границ для поиска синих объектов, в результате чего граница была найдена подбором значений.