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 №2

«Построение карт с использованием SLAM»

Выполнили:

студент группы ИВТВМбд-31

Артюхин Д.С.

Проверила:

Святов К.В.

Ульяновск, 2019

**Содержание**

[1. Задание 3](#_Toc9544347)

[2. Программа V-Rep и её возможности 3](#_Toc9544348)

[3. Метод одновременной локализации и построения карты (SLAM) 6](#_Toc9544349)

[4. Листинг программы 7](#_Toc9544350)

[5. Вывод 9](#_Toc9544351)

# Задание

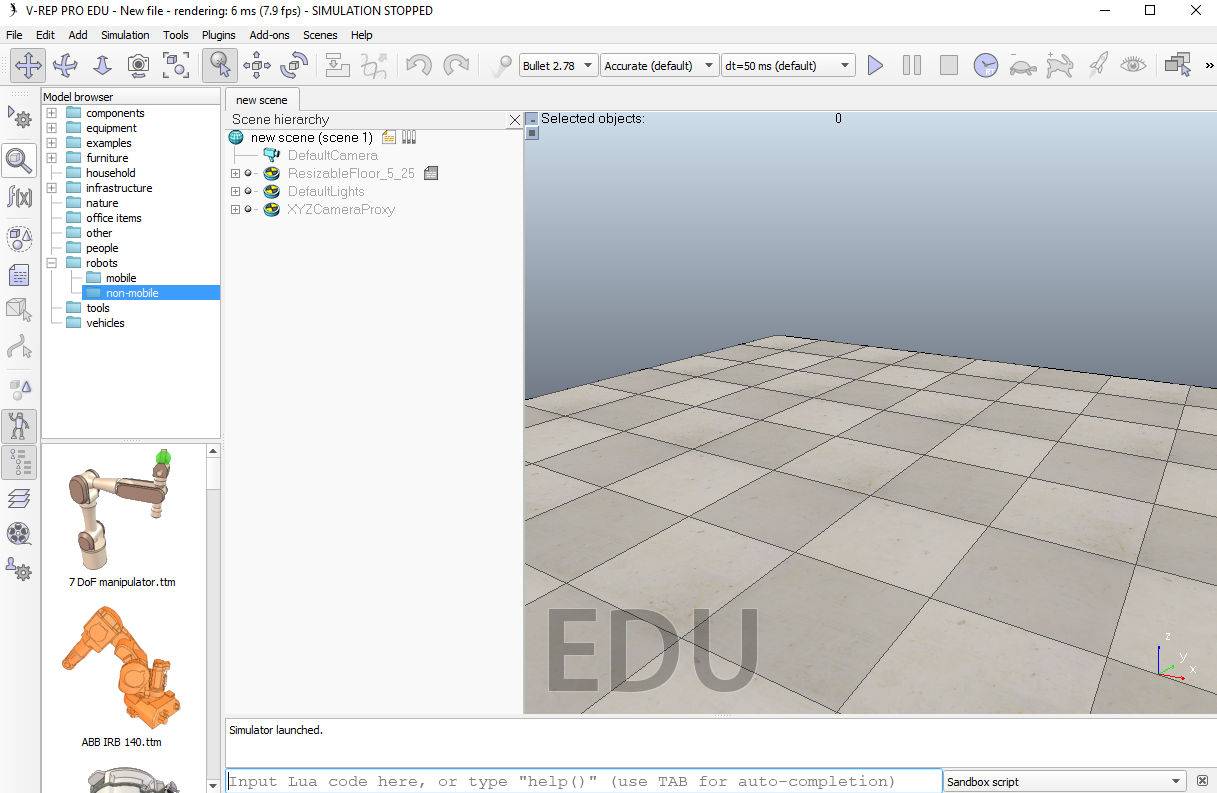
В среде V-REP необходимо с реализовать построение карты окружающего робота пространства с использованием библиотеки любой библиотеки SLAM (например, BreezySLAM) со сценой, разработанной в лабораторной работе №1 в соответствии с вариантом.

Усложненное задание: В среде V-REP Remote API необходимо разработать программу для робота и проверить ее работоспособность в симуляторе V-Rep, обеспечивающую передвижение робота из одной точки в другую (способ указания точки остается на усмотрение студента).

На пути следования робота должны встретиться препятствия. Робот должен объехать их и вернуться на траекторию следования. В качестве возможного способа реализации поставленной задачи можно рассмотреть вариант просчета точек пути следования робота. После объезда препятствия робот стремится к ближайшей точке на пути следования.

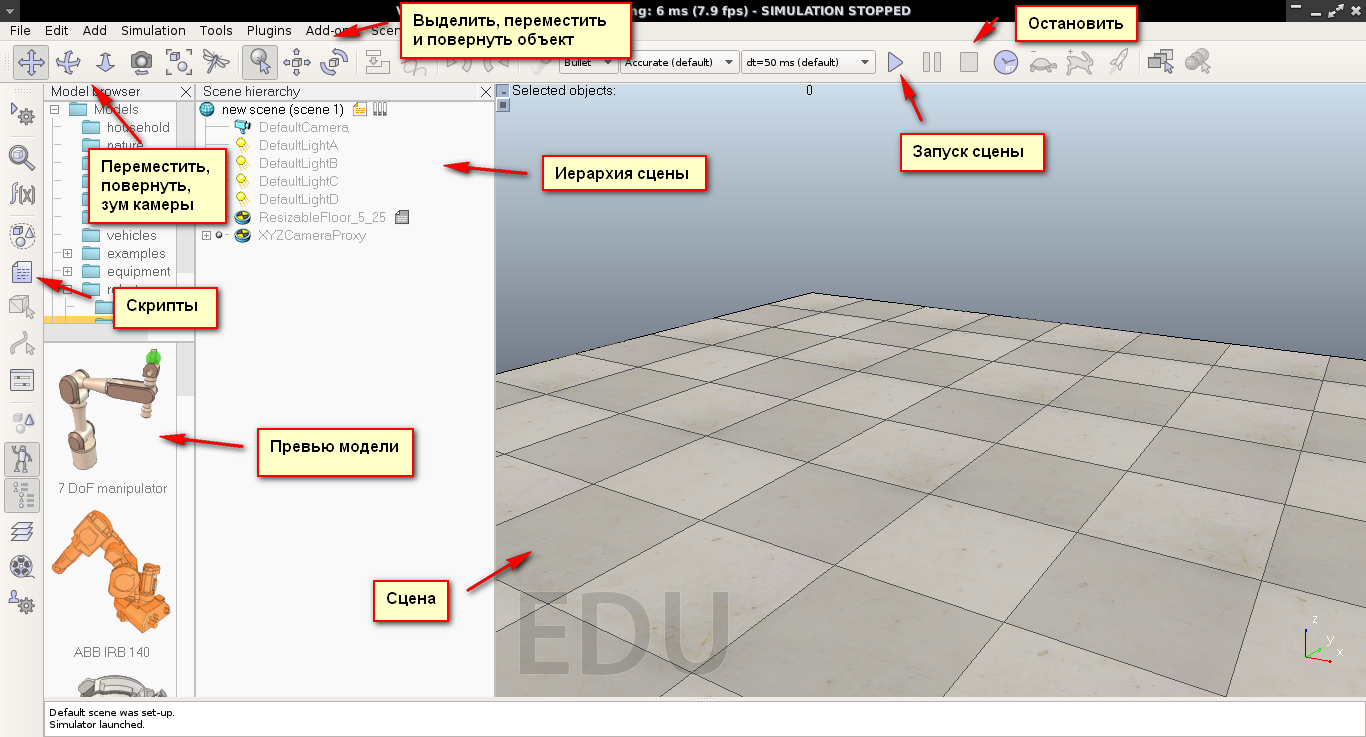
# Программа V-Rep и её возможности

*Рис 2.1 Скриншот программы V-Rep*



**V-REP симулятор** — это результат попыток соответствовать всем требованиям к универсальности и масштабируемости среды моделирования.

*Рис 2.2 Интерфейс программы V-Rep*

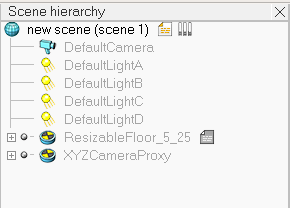
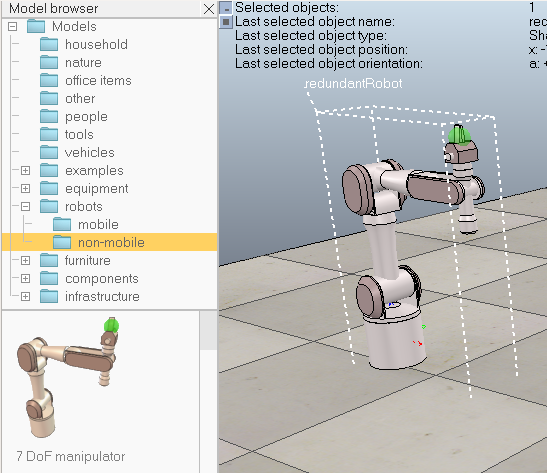


Когда вы открываете среду V-REP, она инициализирует сцену по умолчанию. Пользователь может открывать несколько сцен параллельно, но отображается только текущая.

Панель инструментов содержит набор кнопок, основными из которых являются кнопки для запуска и остановки сцены.  
https://habrastorage.org/files/7df/aa4/4b2/7dfaa44b2dfb4b9f87ac2df22da43921.png

Также инструменты для работы с камерой. Для перемещения и изменения угла просмотра, приближение.  
https://habrastorage.org/files/d9c/105/594/d9c105594e2b4af193e1c7299f80f1c0.png

Для работы с элементами, помещенными на сцену, существуют следующие инструменты: “выделяемость” объекта по нажатию, перемещение и изменение угла расположения объекта. При этом у выделенного объекта появляются дополнительные оси, за которые удобно перемещать и поворачивать.  
https://habrastorage.org/files/9a4/b33/94b/9a4b3394b21e426a8078d26bf55b9e8a.png

Иерархия сцены содержит список открытых сцен. В каждую сцену входят различные объекты, которые могут состоят из целого набора подобъектов, таким образом выстраивая целую иерархию. В сцену по умолчанию входят: Камера, Источники света, Расширяемый пол. По нажатию на плюсик в дереве можно посмотреть зависимые объекты.  
  
  


Также в среде доступны все основные примитивы, добавить которые можно, нажав правую кнопку мыши на сцене и выбрав соответсвующие пункты меню.

# Метод одновременной локализации и построения карты (SLAM)

Метод, используемый в мобильных автономных средствах для построения карты в неизвестном пространстве или для обновления карты в заранее известном пространстве с одновременным контролем текущего местоположения и пройденного пути. Популярные методы приближенного решения данной задачи включают в себя фильтр частиц и расширенный фильтр Калмана. Опубликованные подходы используются в самостоятельном вождении автомобилей, беспилотных летательных аппаратов, автономных подводных аппаратов, планетоходов, домашних роботов и даже внутри человеческого тела.

**Построение карты**

Структурное представление карты местности зависит от среды функционирования робота.

Для выбора наилучшей реализации задач SLAM вводится условная классификация сред функционирования:

* с многочисленными ярко выраженными ориентирами (например, поле с отдельно стоящими кустами)
* с отсутствием ярко выраженных ориентиров (например, коридоры, комнаты, где в качестве ориентиров могут выступать углы)

Если в исследуемой среде нет возможности найти ориентиры, то ее рационально представить в виде массива, где элементы, отражающие положение препятствий, имеют значение 1, а все остальные — 0. (Такое представление карты применяется, например, в алгоритме DP-SLAM)

В случае, когда в исследуемой местности есть многочисленные ориентиры, карта представляет собой массив оценок их местоположения.

Для хранения структуры такой карты проще всего использовать картографическую базу данных, которая отражает положение ориентиров, их уникальные свойства и взаимосвязи. Матрица оценок состояния динамической системы на основе расширенного Фильтра Калмана использует именно этот вариант представления карты.

**Датчики**

В качестве дальномеров в настоящее время используются лазерные дальномеры, гидролокаторы, стереосистемы. Для определения перемещения и поворотов робота могут использоваться одометры.

# Листинг программы

def update(self):

if self.step % settings.steps\_slam == 0:

self.agent.slam(self.bytearray)

self.im = self.to\_image()

self.draw\_agent(self.im)

self.draw\_closest(self.im)

self.draw\_trajectory(self.im)

self.im = cv2.flip(self.im, 0)

self.im = cv2.blur(self.im,(3,3))

cv2.imshow('Simultaneous Localization and Mapping (SLAM)', self.im)

key = cv2.waitKey(1) & 0xFF

self.step += 1

def draw\_closest(self, image):

angles, distances = self.agent.find\_closest()

scale = 30

x = self.agent.pos[0] + int(np.cos((self.agent.theta+angles[0])\*np.pi/180) \* distances[0]\*scale)

y = self.agent.pos[1] + int(np.sin((self.agent.theta+angles[0])\*np.pi/180) \* distances[0]\*scale)

cv2.line(image, (x, y), tuple(self.agent.pos), (150,150,255), 1)

x = self.agent.pos[0] + int(np.cos((self.agent.theta+angles[1])\*np.pi/180) \* distances[1]\*scale)

y = self.agent.pos[1] + int(np.sin((self.agent.theta+angles[1])\*np.pi/180) \* distances[1]\*scale)

cv2.line(image, (x, y), tuple(self.agent.pos), (150,150,255), 1)

def draw\_trajectory(self, image):

for i in range(1, len(self.agent.position\_history)):

color = (255-i\*5,255-i\*5,255-i\*5)

cv2.line(image, self.agent.position\_history[i-1], self.agent.position\_history[i], color, 1)

def to\_image(self):

array = np.frombuffer(self.bytearray, dtype=np.uint8)

gray = np.reshape(array, [settings.image\_size, settings.image\_size])

color = cv2.applyColorMap(gray, self.colormap)

return color

def draw\_agent(self, image):

cv2.circle(image, (self.agent.pos[0], self.agent.pos[1]), self.agent\_radius, self.agent\_color, cv2.FILLED)

# Coordinates of the angle indicator

x = self.agent.pos[0] + int(np.cos(self.agent.theta\*np.pi/180) \* 15)

y = self.agent.pos[1] + int(np.sin(self.agent.theta\*np.pi/180) \* 15)

cv2.circle(self.visited, (self.agent.pos[0], self.agent.pos[1]), self.agent\_radius, (2,2,2), cv2.FILLED)

self.draw\_line\_at\_angle(image, 15, 0, (150,150,150))

self.draw\_line\_at\_angle(image, 15, -135, (150,150,150))

self.draw\_line\_at\_angle(image, 15, 135, (150,150,150))

def draw\_line\_at\_angle(self, im, length, angle, color):

x = self.agent.pos[0] + int(np.cos((self.agent.theta+angle)\*np.pi/180) \* length)

y = self.agent.pos[1] + int(np.sin((self.agent.theta+angle)\*np.pi/180) \* length)

cv2.line(im, (x, y), tuple(self.agent.pos), (150,150,150), 1)

def draw\_sensor\_data(self,im):

data1 = self.agent.read\_lidars()

data = np.array([data1 for \_ in range(20)])

data \*= (255.0/data.max())

offsety = 600

offsetx = 200

for i in range(100):

im[offsetx:offsetx+20,offsety:offsety+len(data1), 0] = data

# Вывод

В данной лабораторной работе я смог реализовать построение карты окружающего робота в пространстве с помощью библиотеке SLAM. Так же научился работать с lidar датчиком и понял его основы в роботостроении.

**Получившийся результат:**

