ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

УПРАВЛЕНИЕ ДИСТАНЦИОННОГО ОБУЧЕНИЯ И ПОВЫШЕНИЯ КВАЛИФИКАЦИИ

Кафедра «Робототехника и мехатроника»

Учебно-методическое пособие

к выполнению лабораторной работы «Передача информации между средами разработки с применением локального интерфейса»
по дисциплине

«Управление мехатронными системами»

Авторы
Лукьянов Е. А.,
Листратенко Я. С.

Ростов-на-Дону, 2018

Аннотация

Лабораторная работа: «Передача информации между средами разработки с применением локального интерфейса» ориентирована на ее выполнение бакалаврами направления подготовки 15.03.06 «Мехатроника и робототехника».

Цель лабораторной работы: изучение возможностей передачи данных между средой моделирования робототехнических систем V-rep и пакетом моделирования Matlab при одновременном моделировании робототехнического или мехатронного устройства. Информационное сопряжение пакетов моделирования осуществляется на основе интерфейса RosInterface.

Авторы

к.т.н., доцент кафедры «Робототехника и мехатроника»
Лукьянов Е. А.,
магистрант группы ММР-21 кафедры «Робототехника и мехатроника»
Листратенко Я. С.

Оглавление

Общие сведения о назначении программных комплексов моделирования и способах их комплексного использования 4

1. Цель, задачи и последовательность выполнения лабораторной работы 4

1.1 Общие сведения об информационном обмене при использовании разных видов интерфейсов 5

1.2 Экспериментальная часть 7

ЗАКЛЮЧЕНИЕ 19

Содержание отчета (пример – в Приложении): 19

Контрольные вопросы 19

Список использованных источников 20

Приложение 20

ОБЩИЕ СВЕДЕНИЯ О НАЗНАЧЕНИИ ПРОГРАММНЫХ КОМПЛЕКСОВ МОДЕЛИРОВАНИЯ И СПОСОБАХ ИХ КОМПЛЕКСНОГО ИСПОЛЬЗОВАНИЯ

Моделирование – это неотъемлемая часть при разработке сложной системы, особенно робототехнической, существует множество программ, которые позволяют создавать виртуальных роботов и исследовать их поведение. Одной из таких программ является среда робототехнического моделирования V-Rep.

V-Rep, представляет собой среду для симулирования различных видов роботов. Среда V-REP предоставляет удобный интерфейс для визуализации действий робота в трёхмерном виртуальном пространстве намного раньше, чем реальный прототип робота будет создан. Однако данный пакет оснащен не слишком широким инструментом для математических расчётов. Для таких задач хорошо подойдет пакет математического моделирования Matlab, обладающий большими возможностями и функционалом для выполнения разнообразных расчетов и анализа. Объединение возможностей этих программ возможно при использовании специальных библиотек, которые позволяют программам взаимодействовать друг, с другом используя сетевой интерфейс.

1. ЦЕЛЬ, ЗАДАЧИ И ПОСЛЕДОВАТЕЛЬНОСТЬ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

Целью является: изучение возможностей передачи данных между средой моделирования робототехнических систем V-rep и пакетом моделирования Matlab при моделировании робототехнического или мехатронного устройства.

Задачи, которые должны быть решены:

Настройка сенсорной (сканирующей) системы виртуального робота и задание ее выходных параметров.

Настройка соединения клиент-сервер, назначения порта, через который будет осуществляться взаимодействие пакетов V-rep и Matlab.

Создание (написание) управляющей программы в Matlab, выполняющей считывание данных о параметрах робота в среде V-rep.

Получение и анализ временных характеристик процесса передачи данных между V-rep и Matlab, таких как «виртуальное

время» исполнения фрагментов управляющих программ в этих пакетах работы и время отклика.

При выполнении ЛАБОРАТОРНОЙ РАБОТЫ рекомендуется придерживаться следующей последовательности:

1. Изучение методов взаимодействия между программами моделирования с использованием локального интерфейса.
2. Создание сцены с объектом, совершающим движение и имеющего сканирующую систему на базе видеосенсора в среде робототехнического моделирование V-rep.
3. Написание программного кода, определяющего логику управления объектами на сцене в программе моделирования V-rep.
4. Настройка интерфейса RosInterface для обмена информацией между средой моделирования V- Rep и Matlab.
5. Создание управляющей программы в пакете моделирования Matlab.
6. Моделирование движения объекта на сцене и передачи информации между пакетами V- Rep и Matlab.
7. Оформление отчета о выполненной лабораторной работы.
8. Подготовка к защите лабораторной работы (см. вопросы в конце методических рекомендаций).
9. Защита лабораторной работы.

1.1 Общие сведения об информационном обмене при использовании разных видов интерфейсов

Для начала необходимо определить, что такое интерфейс[2]. Интерфейс представляет собой комплекс физических и логических форм взаимодействия отдельных компонентов технических средств, входящих в состав операционной системы. Другими словами, это совокупность определенных алгоритмов и соглашений по обмену информацией между компонентами (логический тип интерфейса), а также объединение механических, физических и функциональных характеристик, с помощью которых взаимодействие реализуется (физический тип интерфейса).

Внутримашинный интерфейс представляет собой систему связи и средств соединения блоков и узлов ЭВМ друг с другом. На деле он объединяет в себе электрические линии связи (провода), схему сопряжения с составляющими компьютера, а также прото-

колы (алгоритмы) передачи сигналов. Машинный интерфейс, в свою очередь, подразделяется на односвязный и многосвязный. В первом случае, связь всех блоков ПК друг с другом осуществляется с помощью локальных проводов, а во втором – с помощью общей или системной шины.

Внешний интерфейс – это система связи компьютера с периферийными устройствами или с остальными ЭВМ. Они также подразделяются на несколько типов: интерфейс периферийных устройств и сетевой интерфейс. Первый подключается при помощи шин ввода-вывода, а второй - в рамках одноранговой сети или сети типа клиент-сервер.

Интерфейс применяется везде, где есть взаимодействие между двумя объектами, будь то управление человеком робота или взаимодействие между двумя роботами.

В текущей работе рассмотрим внешний сетевой интерфейс.

Есть два основных способа взаимодействия между программами.

Первый позволяет обходиться без специализированных интерфейсов, но он не всегда удобен для применения. Его суть заключается во взаимодействии с файлами, сохраненными программой. Для понимания принципа работы такого метода рассмотрим взаимодействие между двумя программами. Для того чтобы получить необходимую информацию у первой программы, необходимо записать данные в файл, а со второй программы загрузить данный файл. Главным же минусом такого метода является не возможность управления в «реальном времени» параметрами целевой программы.

Второй же способ позволяет напрямую считывать данные непосредственно из программы, то есть с помощью интерфейса можно сразу отправлять команды и данные. Для этого программы разделяются на программу – клиент и программу – сервер или же раздатчик и подписчик. После чего создается поток данных, из которого уже можно считать необходимую информацию.

Поток данных [5] определяет информацию, передаваемую через некоторое соединение от источника к приемнику. Реальный поток данных может быть информацией, передаваемой по кабелю между двумя устройствами, пересылаемыми по почте письмами, магнитными лентами или дискетами, переносимыми с одного компьютера на другой и т.д. Принцип действия потоковой передачи данных изображен на рисунке 1.

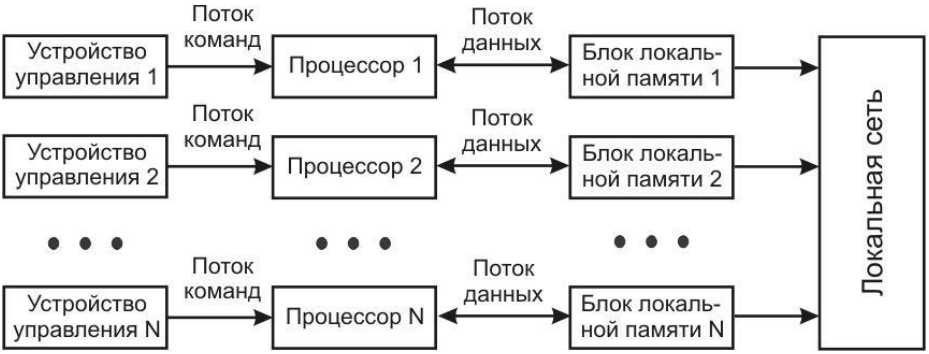

Рисунок 1 – Схема потоковой передачи данных.

Так же помимо передачи данных в локальной сети, всю информацию можно передавать через сеть Internet или по беспроводному каналу связи непосредственно в целевую систему.

Практически при любом взаимодействии двух или более программ используются интерфейсы, важными параметрами которых, являются временные характеристики созданных каналов обмена, поскольку от этого зависит скорость выполнения вычислений и качество синхронизации программ.

В данной лабораторной работе рассмотрим информационное взаимодействие двух пакетов моделирования на примере передачи изображений с виртуального сканирующего устройства и удаленного управления этой системой, и оценим временные задержки канала передачи.

1.2 Экспериментальная часть

Постановка задачи. В ходе проведения экспериментальной части студент должен рассмотреть процесс обмена информацией между программами и его организацию для моделирования динамических систем.

Выходной информацией, полученной в результате работы, будет являться изображение, полученное с видео сенсора в V-rep и отосланное путем потоковой передачи данных с использованием интерфейса[3] в пакет моделирования Matlab.

Программное обеспечение. Для реализации проведения работы необходимо иметь установленный пакет MatLab и средство для моделирования робототехнических систем V-rep.

Ход работы:

Создание объекта, имеющего сканирующую систему.

1. Запустить V-rep. Создать файл сцены. File – NewScene.

2. Добавить на сцену объект – параллелограмм, который будет являться «основанием носителя» сканирующей системы. Для этого в верхней панели инструментов нажать Add – PrimitiveShape – Cuboid. Как показано на рисунке 1. Размеры можно оставить без изменений.

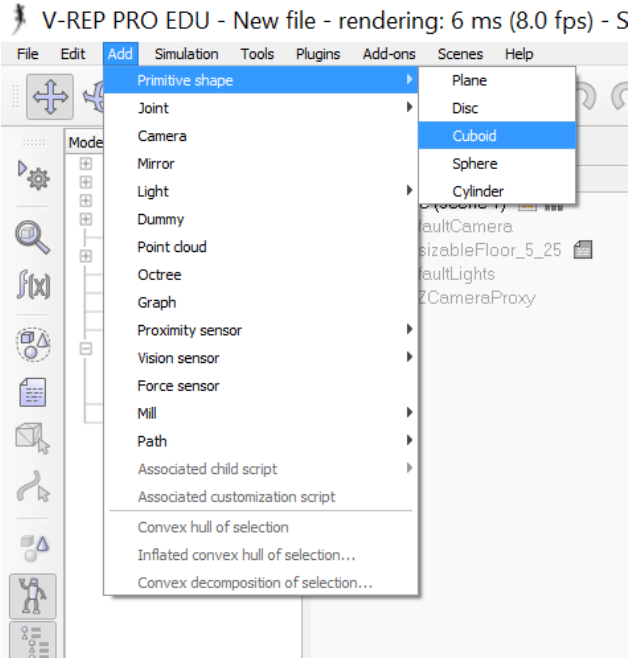

Рисунок 2– Добавление кубоида на сцену.

3. Добавить на сцену двигатель, который позволит сканирующей системе вращаться. Для этого в верхней панели инструментов нажать Add – Joint – Revolute. Разместить данный двигатель в центре Cuboid. Для перемещения используется инструмент Object/Itemshift и в «дереве» сцены выбрать объект Revolute_joint. Далее двигатель необходимо сдвинуть вдоль оси Z. Для этого в появившемся окне выбрать вкладку Position и в строке Z-coord [m] ввести значение +1.0000e-01.

4. Добавить на сцену вторую, подвижную, часть сканера. Для этого в верхней панели инструментов нажать Add – PrimitiveShape – Cuboid. В появившемся окне в строке Z-Size[m] ввести значение +5.0000e-01. Так же необходимо изменить местоположение кубоида, с помощью инструмента Object/Itemshift выбрать Cuboid0 и в появившемся окне выбрать вкладку Position, далее в

строке Z-coord [m] ввести значение +3.5000e-01.

5. Теперь необходимо настроить взаимодействие между этими тремя объектами, т.е. сделать так, что бы они двигались относительно друг друга. Для этого в дереве сцены необходимо переместить объект Revolute_joint на Cuboid, а Cuboid0 на Revolute_joint. В итоге должно получится, как на рисунке

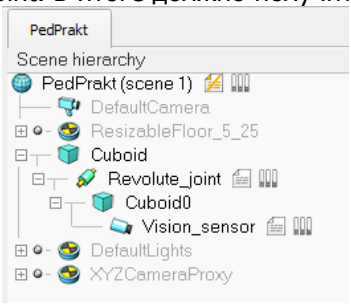

Рисунок 3– Дерево сцены

6. Открыть свойства Cuboid0, для этого в дереве сцены сделать двойной щелчок мыши на изображении куба, рядом с названием объекта. В появившемся окне нажать на кнопку Showdynamicpropertiesdialog. Далее снять галочки с BodyisrespondableиBodyisdynamic.

7. Добавить сенсор. Для этого в панели инструментов нажать Add – Visionsensor – Perspectivetype. Как показано на рисунке В дереве сцены Visionsensor переместить на Cuboid0.

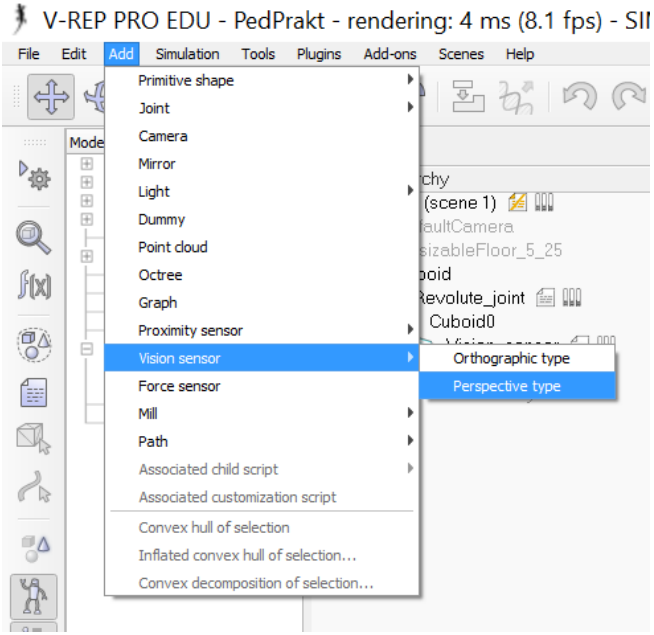

Рисунок 4– Добавление видео сенсора

Для перемещения сенсора нужное положение с помощью инструмента Object/Itemshift выбрать Visionsensor и в появившемся окне выбрать вкладку Position, далее в строке Z-coord [m] ввести значение +5.0000e-01.

Изменить угол наклона сенсора.с помощью инструмента Object/Itemrotate выбрать Visionsensor. В появившемся окне выбрать вкладку Rotation и в строке Alpha ввести значение +1.2000e+02.

8. Добавить скрипты к объектам. Для этого в левой панели инструментов нажать Scripts. В появившемся окне нажать Insert-newscript. В строке Scripttype выбрать Childscript (non-threaded).

В списке скриптов выбрать созданный скрипт и в строке Associatedobject выбрать Revolute_joint.

В итоге должно быть показано, как на рисунке

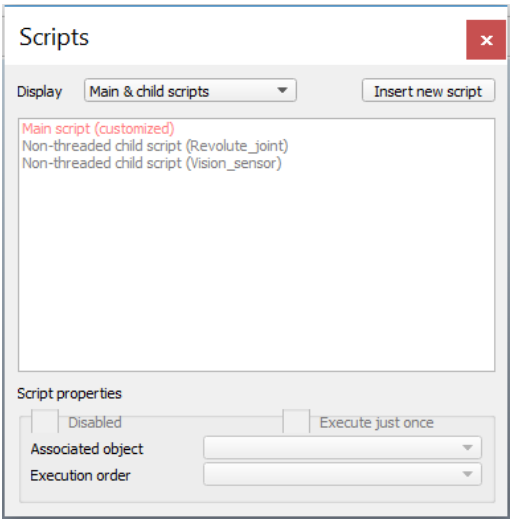

Рисунок 5 – Панель скриптов

Повторить то же самое, только для нового скрипта в строке Associatedobject выбрать Visionsensor.

Сканер полностью готов. В результате должна получиться система, показанная на рисунке

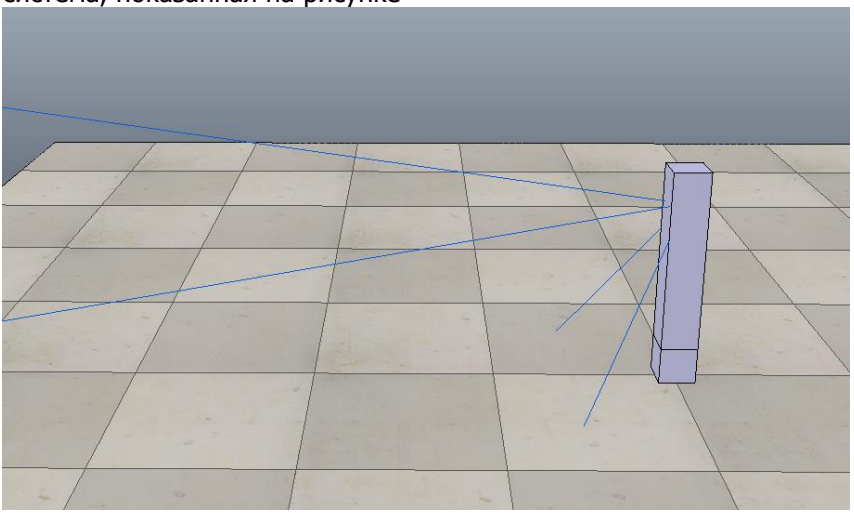

Рисунок 6 – Готовая сцена

Основные алгоритмы V-rep.

Для того, чтобы заставить сканер вращаться, необходимо в ассоциированный с объектом Revolute_joint скрипте написать программу, управляющую этим движением. Для этого необходимо

открыть скрипт, который находится в дереве сцены, это показано на рисунке, напротив названия объекта. В появившемся окне редактора написать следующий код:

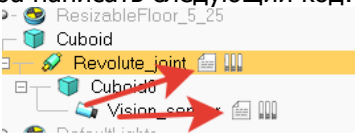

Рисунок 7 – Иконки объектов сцены и их иерархических связей.

```
if (sim_call_type==sim_childscriptcall_initialization) then
Join-
tHandle=simGetObjectAssociatedWithScript(sim_handle_self)
end

if (sim_call_type==sim_childscriptcall_actuation) then
local t=simGetSimulationTime()
simSetJointPosi-
tion(JointHandle,90*math.pi*math.sin(0.25*t)/180)
end
```

В итоге окно скрипта должно выгладить так, как показано на рисунке

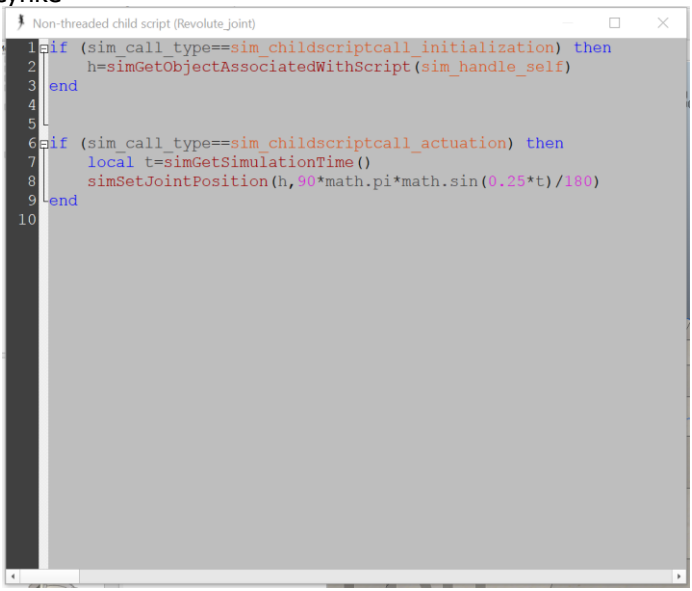

Рисунок 8 – Редактор скриптов

Условие sim_call_type==sim_childscriptcall_initialization, проверяет, был ли произведен запуск скрипта. Данное условие возвращает истину только один раз при запуске, следовательно, выполняется только один раз.

JointHandle=simGetObjectAssociatedWithScript(sim_handle_self)

Переменной JointHandle присваивается ссылка на объект sim_handle_self, то есть на Revolute_joint. Так как данная функция ссылается на себя.

Условиеsim_call_type==sim_childscriptcall_actuation вызывается при каждом кадре, поэтому все описанное в теле условия повторяется постоянно на протяжении всей работы скрипта.

localt=simGetSimulationTime() – локальной переменной t присваивается время симуляции от момента запуска скрипта. Измеряется в миллисекундах.

simSetJointPosition(JointHandle,90*math.pi*math.sin(0.25*t)/180) – данная функция задает позицию объекта Revolute_joint. Принимает в себя ссылку на объект JointHandle и позицию. Это позволяет объекту вращаться на 180 градусов.

Настройка видео сенсора.

Открываем скрипт Visionsensor.

```
if (sim_call_type==sim_childscriptcall_initialization) then
  -- Получаем ассоциации с объектом
VisionSensor=simGetObjectHandle('Vision_sensor')

-- Активируем систему клиент – сервер
pub=simExtRosInterface_advertise('/image', 'sensor_msgs/Image')
simExtRosInterface_publisherTreatUInt8ArrayAsString(pub)
sub=simExtRosInterface_subscribe('/image', 'sensor_msgs/Image', 'imageMessage_callback')
simExtRosInterface_subscriberTreatUInt8ArrayAsString(sub)
end

if (sim_call_type==sim_childscriptcall_sensing) then
-- Публикация изображения с видео сенсора. В массив d входят все необходимые параметры, такие как : Разрешение изображения, кодировка и.т.д.
localdata,w,h=simGetVisionSensorCharImage(VisionSensor)
d={}
d['header']={seq=0,stamp=simExtRosInterface_getTime(), frame_id="a"}
```

```
d['height']=h
d['width']=w
d['encoding']='rgb8'
d['is_bigendian']=1
d['step']=w*3
d['data']=data
simExtRosInterface_publish(pub,d)
end

if (sim_call_type==sim_childscriptcall_cleanup) then
-- Выключение режима клиент - сервер.
simExtRosInterface_shutdownPublisher(pub)
simExtRosInterface_shutdownSubscriber(sub)
end
```

Система управления в пакете Matlab.

Для управления и считывания данных из пакетаV-rep используется пакет моделирования Matlab. Управление осуществляется при помощи библиотеки remoteApi.dll. Данная библиотека содержит в себе набор функций, которые позволяют устанавливать локальное соединение между программами.

Для начала необходимо подготовить программу к работе.

Ход работы:

1. Открыть Matlab
2. Создать новый скрипт. В верхней панели инструментов нажать NewScript это показано на рисунке.

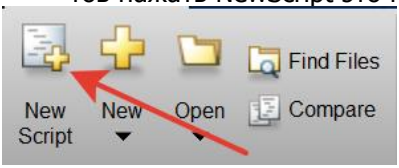

Рисунок 9 – Панель управления

3. Сохранить скрипт. В верхней панели инструментов нажать Save. Далее указать путь и название.
4. Открыть папку с сохраненным скриптом. И скопировать в нее файл remoteApi.dll, который находятся в папке V-rep, по пути:V-REP3\V-REP_PRO_EDU\programming\remoteApiBindings\lib\lib\, в зависимости от разрядности системы выбрать 64bit или 32bit.

5. Открыть папку в V-rep, которая находиться по пути:V-REP3\V-REP_PRO_EDU\programming\remoteApiBindings\matlab\matlab и скопировать файлы remApi.m и remoteApiProto.m в папку с сохраненным скриптом.
6. Перейти к Matlab. В окно скрипта добавить следующий код.

Подготовка к соединению. Подключение библиотеки remoteApi.dll, установка локального соединения.

```
vrep=remApi('remoteApi'); % используется (remoteApiProto.m)
vrep.simxFinish(-1); % Закрытие всех соединений
clientID=vrep.simxStart('127.0.0.1',19997,true,true,5000,5);
disp('Программа запущена');
```

Функция simxFinish с аргументом -1 прекращает любые соединения.

Функция simxStart производит установку соединения с удаленным сервером, но так как V-rep и Matlab находятся на одном компьютере, используется локальный адрес. Входящие аргументы.

connectionAddress: IP адрес, где сервер находится (тоесть адрес сервера с V-REP) 127.0.0.1 – локальный адрес компьютера.

connectionPort: Порт, с которым происходит соединение.

waitUntilConnected: Если True, тогда функция блокируется на время соединения или пока не закончится время тайм-аута.

doNotReconnectOnceDisconnected: Если True, тогда коммуникационный поток не будет пытаться переподсоединиться в случае потери соединения.

timeOutInMs: Тайм-аут соединения в милисекундах (для первого соединения).

commThreadCycleInMs: Указывает на то, как часто должен происходить обмен пакетами данных. Уменьшение этого числа повышает чувствительность, а в роли значения по умолчанию рекомендуется 5.

Далее производится проверка на успешность соединения. В данном случае условие выполняется только в том случае, когда переменной clientID было возвращено значение IDсоединения. А это возможно только в том случае, если было установлено соединение.

```
if (clientID>-1)
disp('Соединение установлено');
[camerro,vision1]          =
```

vrep.simxGetObjectHandle(clientID,'Vision_sensor',vrep.simx_opmode_oneshot_wait);

Функция simxGetObjectHandle получает ссылку на видео сенсор.

vrep.simxStartSimulation(clientID,vrep.simx_opmode_oneshot_wait);

Функция simxStartSimulation позволяет удаленно запустить симуляцию сцены V-rep.

vrep.simxGetVisionSensorImage2(clientID,vision1,0,vrep.simx_opmode_streaming);

ФункцияsimxGetVisionSensorImage2 с аргументом vrep.simx_opmode_streaming инициализирует поток данных с видеосенсора для передачи.

Цикл, начальное значение 0, конечное 29. В теле цикла происходит считывание данных с потока и возвращение изображений.

```
i=0
whilei<30
[returnCode,resolut,image]=
vrep.simxGetVisionSensorImage2(clientID,vision1,0,vrep.simx_opmode_buffer); % Извлечение потоковых данных
```

Функция simxGetVisionSensorImage2 с аргументом vrep.simx_opmode_buffer возвращает изображения и параметры, указывающие на успешность возвращения.

```
if (returnCode==vrep.simx_return_ok)
```

Проверка возвращаемых параметров, если считывание изображения успешно, то показываем текущее изображение. (В данном примере изображения хранятся в переменной image и постоянно перезаписываются, но есть так же возможность создать массив изображений, для дальнейшей работы с ними.)

```
imshow(image);
pause(0.1)
```

Пауза нужна для замедления цикла, в противном случае весь цикл пройдет за доли секунды.

```
i=i+1
```

Операция инкремента.

```
end
end
vrep.simxStopSimulation(clientID,vrep.simx_opmode_oneshot_wait);
```

Функция simxStopSimulation останавливает симуляцию сце-

ны V-rep.

vrep.simxFinish(clientID);

Функция simxFinish завершает соединение с ID = clientID.

else

disp('Ошибка соединения');

Это сообщение вызывается тогда, когда соединение не удалось установить, то есть функция vrep.simxStart не смогла возвратить clientID.

end

vrep.delete();

Прерывает любые потоки данных между программами.

disp('Программа завершена');

Пример окна со скриптом показан на рисунке

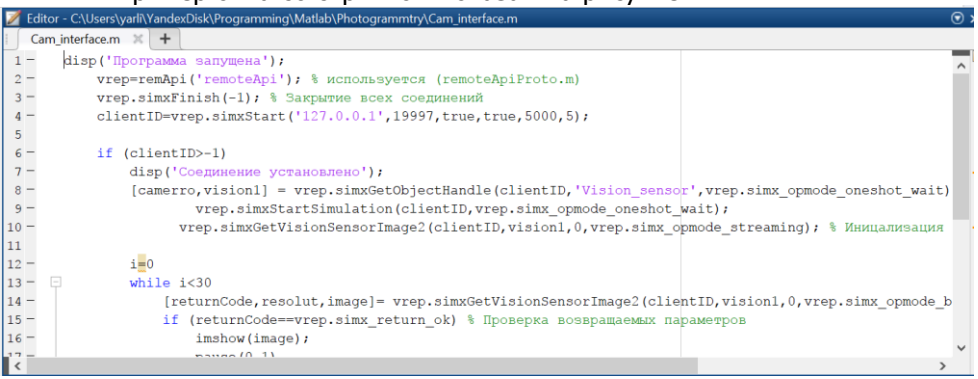

Рисунок 10 – Редактор скриптов Matlab

Для запуска моделирования в верхней панели инструментов нажать Run, как показано на рисунке

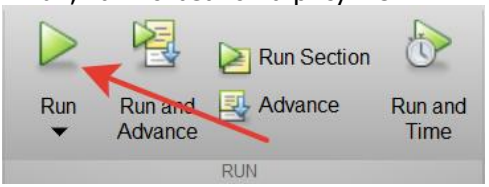

Рисунок 11 – Панель управления с кнопкой запуска Matlab.

После установки соединения сцена в среде V-rep автоматически запустится и начнет передавать данные в Matlab. В сою очередь в пакете Matlab появится окно с изображением, считанным с видео сенсора, с интервалом в 0.1 сек, что показано на рисунке 2.

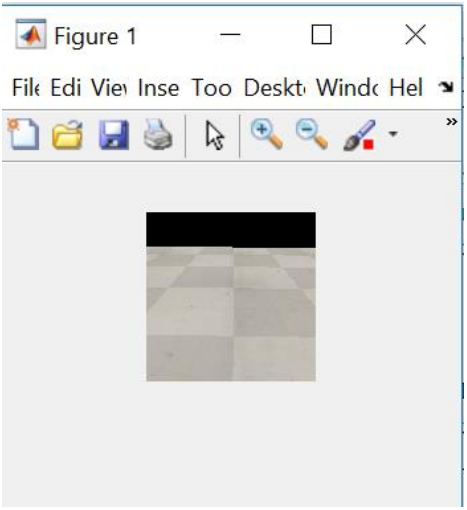

Рисунок 12 – Изображение, полученное с видео сенсора.

Для того, чтобы получить время задержки симуляции между Matlab и V-Rep необходимо воспользоваться функцией vrep.simxGetLastCmdTime, которая позволяет получить время исполнения команды во V-Rep.

Для этого присвоим время симуляции переменной: VrepTime=vrep.simxGetLastCmdTime(clientID);

Далее при помощи функций tic toc, можно найти время работы конкретного участка кода.

Присвоим переменной MatlabTime значение задержки.

MatlabTime = toc;

А так же находим задержку.

VrepTime = VrepTime/ 1000;

Diff = abs(MatlabTime - VrepTime);

В итоге получаем разницу во времени при выполнении участка кода.

Выводим значение на экран:

Info = ['Время симуляции V-rep = ',num2str(VrepTime),' с. Время симуляции Matlab = ', num2str(MatlabTime), ' с. Время, затраченное на отправку данных = ',num2str(Diff), ' с.'];

disp(Info);

При помощи библиотеки remoteAPI.dll можно получать и передавать практически любые параметры. Примеров использования данного интерфейса множество. Так как среда робототехнического не обладает мощным инструментарием для матема-

тических расчетов, для этого и используют пакеты, подобные Matlab, например, для расчета моментов двигателя, который используется уже непосредственно в симуляции. Так же разработчик интерфейса предусмотрел работу с такими языками программирования, как python, java, lua, octave и urbi.

ЗАКЛЮЧЕНИЕ

В данной работе были рассмотрены методы информационного взаимодействия между программами VREP и Matlab в процессе моделировании робототехнических систем. Показаны возможности удаленного управления программой и передачи сенсорных данных из пакета VREP в пакет Matlab для их обработки (распознавания) и расчета управлений для виртуального объекта.

СОДЕРЖАНИЕ ОТЧЕТА (ПРИМЕР – В ПРИЛОЖЕНИИ):

- Цель работы;
- Задачи, решенные при выполнении лабораторной работы;
- Описание метода взаимодействия между пакетами моделирования.
- Заключение о результатах выполненных работ и экспериментальной оценки временных параметров информационного взаимодействия пакетов VREP и Matlab.
- Вывод.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое интерфейс?
2. Для чего нужна библиотека RemoteAPI.dll?
3. Что такое поток данных?
4. В чем заключается особенность работы режима Клиент – Сервер?
5. Как можно изменить интервал считывания изображения?
6. Какой аргумент должна принять функция simxGetVisionSensorImage2, что бы данная функция возвращала изображение?
7. Как получить ассоциации с объектом?
8. Что записывается в переменную clientID?
9. Какая функция отвечает за получение информации о времени симуляции участка кода.

10. Есть ли задержка при передаче данных межу клиентом и сервером?

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Мячев, А. А. Интерфейсы систем обработки данных / А.А. Мячев, В.Н.Степанов, В.К. Щербо. - М.: Радио и связь, 1989. - 416 с.
2. Портянкин, Иван Swing. Эффектные пользовательские интерфейсы / Иван Портянкин. - М.: ЛОРИ, 2011. - 608 с.
3. Эрглис, К. Э. Интерфейсы открытых систем. Учебный курс / К.Э. Эрглис. - М.: Горячая линия - Телеком, 2000. - 256 с.
4. Вирт, Н. Алгоритмы и структуры данных / Н. Вирт. - М.: [не указано], 1989. - 568 с.

ПРИЛОЖЕНИЕ

Образец оформления протокола лабораторной работы

Лабораторная работа по дисциплине Управление мехатронными системами на тему: Передача информации между средами разработки с применением локального интерфейса.

Выполнил:

Проверил:

ДГТУ, Ростов-на-Дону

20__г.

Цель работы и задачи: _____

Основные функции, использованные в управлении симуляцией. Функция установки порта соединения, функции запуска и остановки симуляции удаленной программы, функция считывания времени симуляции, функция инициализации потока данных с видеосенсора: _____

Результаты расчета временных задержек и нахождение среднего значения: _____

Вывод: _____