Приложение Б. Инструкция по использованию ПО для участников

А. Состав программного обеспечения

В виртуальной машине установлены

1. OS Linux Ubuntu 16.04
2. docker container с необходимым для решения задач ПО
    a. CoppeliaSim 4
    Путь /opt/csim/
    b. ROS kinetic
    Путь /opt/ros/kinetic/
    c. Библиотека для работы линейной алгеброй (C++)
    /usr/include/eigen3
    d. VISP
    /usr/include/x86_64-linux-gnu/visp
    e. Библиотеки для python2
    numpy, scipy, cv2
    f. Прочие удобные утилиты
        i. Файловый менеджер в терминале
        mc
        ii. Текстовые редакторы
        gedit, nano, vim
3. Папки с “заготовками” исходного кода, для решения задач
    Папка /home/human/ *из Ubuntu “проброшена” в docker* /home/root/
    a. В Ubuntu
    Бакалавриат /home/human/catkin_ws/src/bac_task
    Магистратура /home/human/catkin_ws/src/mag_task
    b. В docker
    Бакалавриат /home/root/catkin_ws/src/bac_task
    Магистратура /home/root/catkin_ws/src/mag_task
4. Структура “заготовок” исходного кода
    a. Бакалавриат (ROS-package со стандартной структурой)
    (все файлы, кроме main_solve.py (.cpp), *запрещены к редактированию)*
        i. scenes/bac_scene.ttt - Сцена в симуляторе CoppeliaSim, включающая наземного робота Robotino и квадрокоптер.
        ii. src/robotino_model.py - Нода, реализующая кинематическую модель наземного робота Robotino
        iii. src/robotino_trajectory_generator.py - Нода, реализующая генерацию траектории с заданными параметрами
        iv. src/trajectory_drawer.py - Нода, реализующая визуализацию траектории и систем координат сцены (работает совместно с rviz)
        v. src/marker_detector_node.cpp - Нода, реализующая поиск April кода, и извлечение из изображения признаков (features)
        vi. **main_solve.py (.cpp) - Нода, содержит “заготовку” исходного кода для решения задачи**
    b. Магистратура
    (файлы i и ii запрещены к редактированию)
        i. scenes/mag_scene.ttt - Сцена в симуляторе CoppeliaSim, включающая манипулятор, столы, цилиндры и проч.
        ii. ur_aux.py - Модуль содержит функции, необходимые для решения задачи
        iii. **ur_main.py - Модуль содержит “заготовку” исходного кода для решения задачи**

Б. Формирование оценочного балла

Таблица 1. Критерии оценки решения задачи для бакалавриата

Основной критерий, **50 б.**	**Расстояние**, пройденное БПЛА, в лучшей из тестовых попыток. *При условии соблюдения правил и ограничений спецификации задания.* *Максимальное время тестовой попытки 10 минут.* *Количество попыток не более 10.*	
Вспомогательный критерий, **50 б.**	20 б.	● сложность алгоритма ● учет динамики ● фильтрация измерений ● учет ограничений и подобные.
	20 б.	Рецензирование кода (Code review)
	10 б.	Оригинальность решения

Таблица 2. Критерии оценки решения задачи для магистратуры/специалитета

Основной критерий, **50 б.**	**Число успешно вставленных цилиндров в отверстия.** *При условии соблюдения правил и ограничений спецификации задания.* *Максимальное время тестовой попытки 10 минут.* *Количество попыток не более 10.*	
Вспомогательный критерий, **50 б.**	20 б.	● сложность алгоритма ● учет динамики ● фильтрация измерений ● учет ограничений и подобные.
	20 б.	Рецензирование кода (Code review)
	10 б.	Оригинальность решения

В. Некоторые замечания по работе с Ubuntu + docker

1. **Сведения о пользователе Linux Ubuntu**
   User: human
   Password: 1
   Host: host
2. **Важные папки**
   a. /home/human - папка, доступная из docker контейнера
   b. /home/human/catkin_ws/src/bac_task - ROS-пакет с заданием для бакалавриата
      i. ./src/main_solve.py - файл, в котором нужно написать код в указанных местах, решающий задачу.

ii. Редактирование остальных файлов в пакете ЗАПРЕЩЕНО.

c. /home/human/catkin_ws/src/mag_task - файлы для решения задачи магистратуры
   i. ./ur_main.py - файл, в который нужно написать код, решающий задачу.
   ii. Редактирование остальных файлов в пакете ЗАПРЕЩЕНО.
d. /home/human/Desktop - рабочий стол. Содержит два файла для работы с docker контейнером
   i. ./run_docker.bash - запускает docker контейнер и открывает терминал внутри него
   ii. ./exec_docker.bash - запускает дополнительный терминал в docker контейнере. Можно вызывать столько раз, сколько нужно дополнительный терминалов внутри УЖЕ ЗАПУЩЕННОГО docker контейнера

3. Работа с ПО в процессе решения задачи (БАКАЛАВРИАТ)

a. Открыть терминал (Ctrl+Alt+T)
b. Открыть файл для решения
   gedit /home/human/catkin_ws/src/bac_task/main_solve.py
c. Вписать свое решение. Сохранить
d. Запустить docker контейнер
   sudo ./Desktop/run_docker.bash
   Ввести пароль “1”
e. Скомпилировать catkin_ws
   roscd && cd .. && catkin build
f. Запустить симулятор
   ~/catkin_ws/src/bac_task/start_scene.**bash**
g. Запустить roslaunch файл для инициализации сцены и необходимых нод
   roslaunch bac_task init_scene.**launch**
h. Запустить симуляцию (Кнопка “Play”)
i. Запустить дополнительный терминал в docker контейнере
   sudo ./Desktop/exec_docker.bash
j. Запустить ваше решение (**ВНИМАНИЕ**! Запускается не тот файл, в котором решение)
   python /home/human/catkin_ws/src/bac_task/**main_solve_wrapper**.py
k. Запустить дополнительный терминал в docker контейнере
   sudo ./Desktop/exec_docker.bash
l. Вызвать rosservice, который запустит роботов
   rosservice call /start_robots “data: true”
m. Оценить работоспособность роботов в соответствии со спецификацией задания
n. Остановить симуляцию. (Кнопка “Stop”)
o. Завершить работу вашего решения (Ctrl+C)
p. Внести правки в код /home/human/catkin_ws/src/bac_task/main_solve.py
q. Повторять (пп. 3.*g* - *3.p)* до тех пор, пока задача не будет решена наилучшим образом
r. *Для более точной отладки регуляторов, возможен запуск ПО без запуска графической части симулятора. Для этого необходимо в пункте 3.f запускать другой roslaunch файл
   roslaunch bac_task start_scene_headless.**bash**
   А для отладки пользоваться утилитой построения графиков
   rqt_plot
s. ** Для запуска тестирования (вывод пройденного расстояния)
   rosrun bac_task judge

4. Работа с ПО в процессе решения задачи (МАГИСТРАТУРА)

a. Открыть терминал (Ctrl+Alt+T)
b. Открыть файл для решения
   gedit /home/human/catkin_ws/src/mag_task/ur_main.py
c. Используя заготовленные функции из файла ur_aux.py решить задачу и вписать свое решение. Сохранить
d. Запустить docker контейнер
   sudo ./Desktop/run_docker.bash
   Ввести пароль “1”
e. Запустить симулятор
   /opt/csim/coppeliaSim.sh
f. Запустить симуляцию (Кнопка “Play”)
g. Запустить решение
   python /home/human/catkin_ws/src/mag_task/ur_main.py
h. Оценить работоспособность решения
i. Остановить симуляцию (Кнопка “Stop”)
j. Завершить работу вашего решения (Ctrl+C)
k. Внести правки в код /home/human/catkin_ws/src/mag_task/ur_main.py
l. Повторять (пп. 4.*g - 4.k)* до тех пор, пока задача не будет решена наилучшим образом

5. Рисунки, поясняющие расположение систем координат в сценах

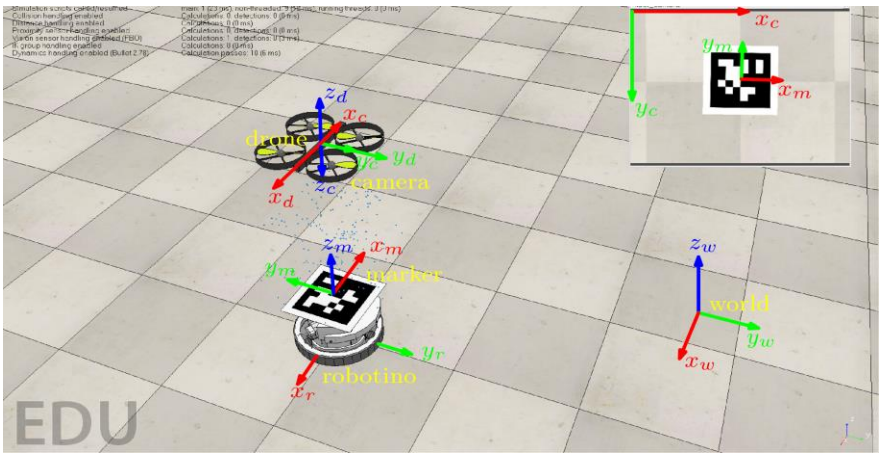

Рисунок 1. Задача для бакалавриата

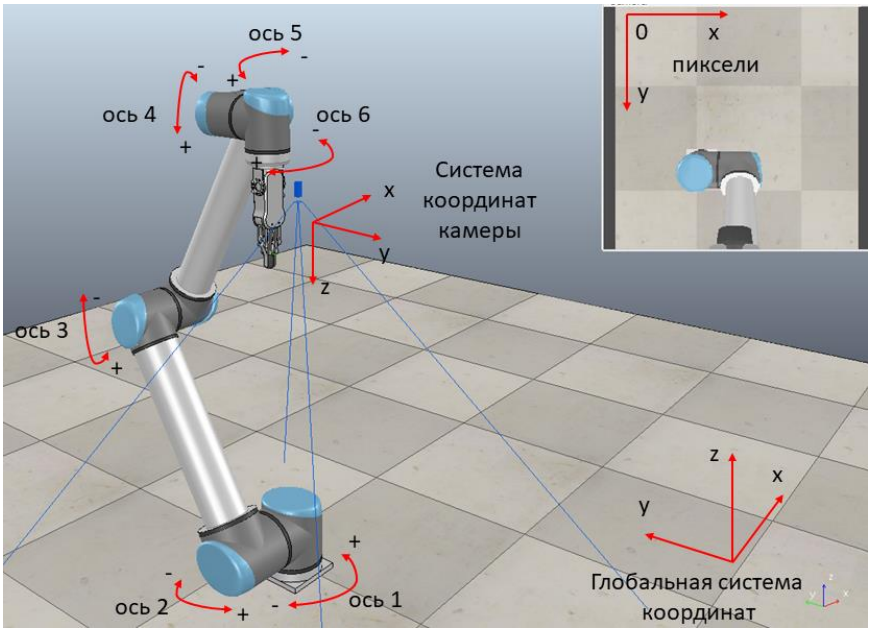

Рисунок 2. Задача для магистратуры