

Я-Профессионал
Робототехника

Вебинар 1

Спикеры: Кирилл Артемов, Сергей Колюбин

Университет ИТМО

24 января 2020

Вводная часть

Вебинар состоит из трех частей

I. Организационная информация

- ▶ Важные даты и координаты
- ▶ Структура и содержание заданий
- ▶ Система оценивания: квалификация в суперфинал, личный и командный зачет

II. Теоретический минимум

- ▶ Преобразования координат
- ▶ Прямая и обратная задачи кинематики
- ▶ Планирование траекторий
- ▶ Кинематика и динамика роботов
- ▶ Управление движением роботов. Визуальная обратная связь

III. Кик-старт по программному обеспечению

- ▶ Linux Ubuntu
- ▶ Установка и настройка docker
- ▶ Robot Operation System (ROS)
- ▶ CoppeliaSim (бывший V-REP)
- ▶ Доступные для использования библиотеки (OpenCV, VISP, eigen, numpy, scipy)

Важные даты и координаты

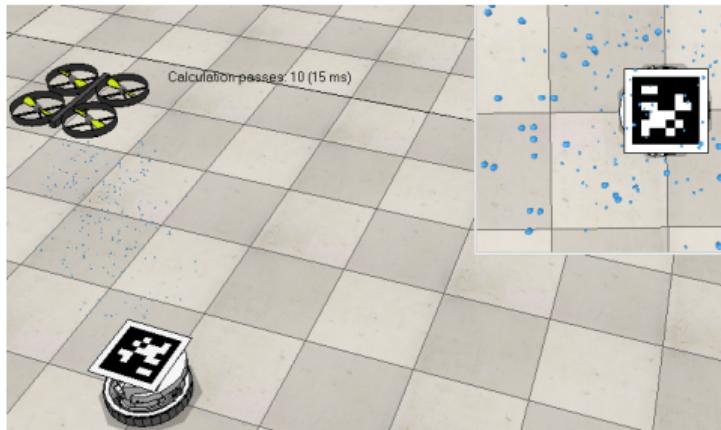
Первый тур (региональный финал, виртуальный тур)

- 15 февраля
- Москва, СПб, Иннополис, Екатеринбург - старт в 10:00 по московскому времени
- Томск, Владивосток - старт в 7:00 по московскому времени
- адреса площадок на https://yandex.ru/profi/second_stage
- продолжительность до 6 астрономических часов (360 минут)

Второй тур (суперфинал, практический тур)

- с 28 февраля по 1 марта
- Центр НТИ по робототехнике, Иннополис
- Питание и проживание бесплатно
- Проводится в два дня: 1 пробная и 2 основных попытки
- Командная работа

Задание: бакалавры



Задача – колесная омниплатформа движется по заданной траектории (восьмерка, только одометрия), квадрокоптер летит над ней, ориентируясь бортовой камерой по визуальному маркеру

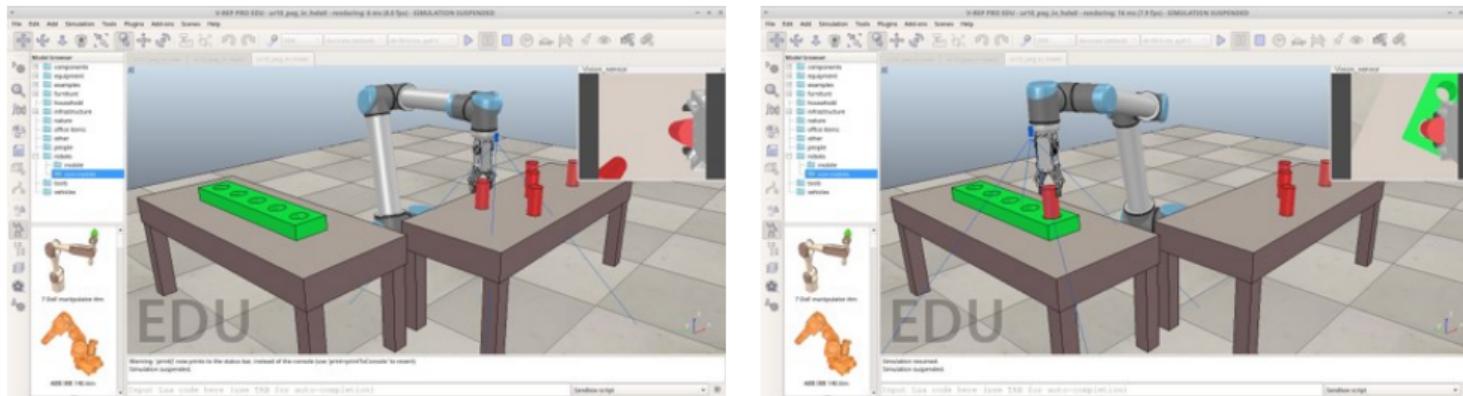
Решение – программа на C++ или Python с апробацией в симуляторе, индивидуальная работа

Задание: бакалавры

Критерии

- ▶ движение с заданной точностью для омниплатформы и квадрокоптера
- ▶ суммарное расстояние, пройденное квадрокоптером по траектории за 10 минут – макс. значения среди всех участников - 50 баллов, остальные пропорционально
- ▶ сложность использованных моделей и алгоритмов
- ▶ качество проработки программного кода
- ▶ оригинальность представленного решения

Задание: магистры/специалисты



Задача – манипулятор с eye-in-hand камерой и силомоментным датчиком на фланце вставляет цилиндры в заданные отверстия на паллете
Решение – программа на C++ или Python с апробацией в симуляторе, индивидуальная работа

Задание: магистры/специалисты

Критерии

- ▶ число успешно вставленных в отверстие цилиндров, деленное на время выполнения задания в минутах (полное время до 10 мин.) – макс. значения среди всех участников - 50 баллов, остальные пропорционально
- ▶ заклинившие и вставленные ненадежно цилинды не засчитываются
- ▶ сложность использованных моделей и алгоритмов
- ▶ качество проработки программного кода
- ▶ оригинальность представленного решения

Общая система оценивания

Условия квалификации (допуск во 2 тур)

- ▶ бакалавры – оба робота проходят в соответствии с правилами задания и ограничениями любое ненулевое расстояние
- ▶ магистры/специалисты – вставка хотя бы одного цилиндра в произвольное отверстие на паллете за полное время, отводимое на попытку

Задача на 2 тур – программно-аппаратное решение (C++ или Python) с аprobацией на реальном робототехническом комплексе, командная работа

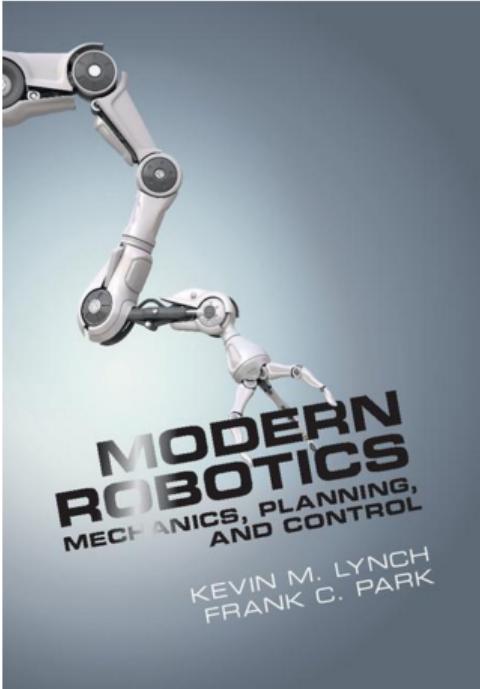
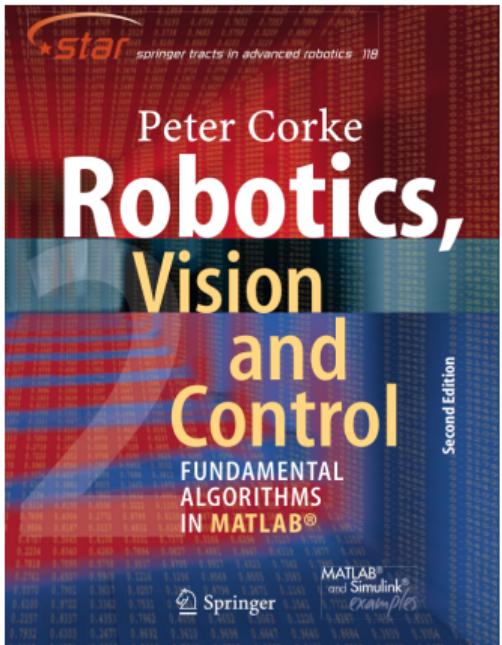
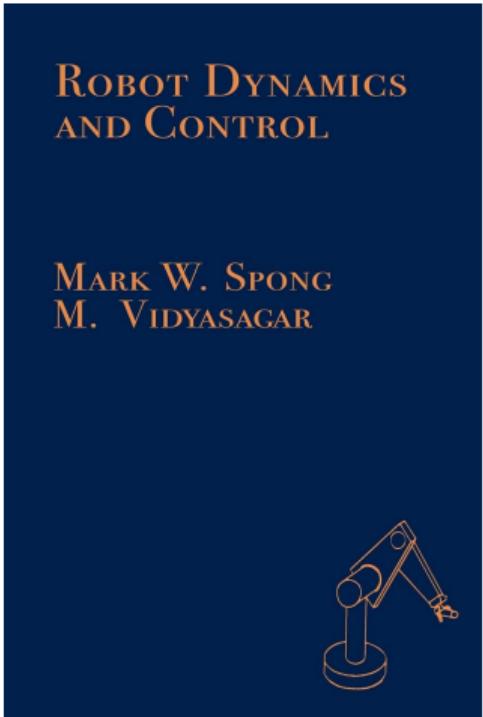
До 100 баллов в каждом туре

Итоговый результат финала (макс. 200 баллов) = личный зачет (1 тур) + командный зачет (2 тур)

Подробный регламент, демо-версии заданий и спецификации на

https://yandex.ru/profi/second_stage

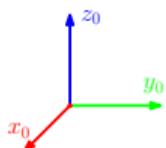
Основная литература



Основные термины и обозначения

Некоторые из принятых обозначений и сокращений в презентации

- ▶ СК — правая система координат
- ▶ Изображения СК



- ▶ O_B — обозначение центра СК B

- ▶ q — вектор обобщенных координат
- ▶ M, C, g — матрицы инерции, кориолисовых и центробежных сил и вектор гравитации
- ▶ $\mathbf{0}$ — нулевая квадратная матрица
- ▶ \mathbf{E} — единичная квадратная матрица
- ▶ ${}^i r_j = [x, y, z, \varphi, \psi, \theta]^T$ — вектор, задающий положение в пространстве
- ▶ ${}^i H_j$ — матрица однородных преобразований. Смещение и поворот j -ой СК относительно i -ой

Параметризация. Основные факты

Положение на плоскости (2D)

- ▶ линейные координаты (x, y)

- ▶ угол θ

- ▶ Положение СК базы робота O_B выражается относительно заданной (нулевой) СК O_0
- ▶ Все СК "внутри" робота (O_E и др.) выражаются относительно его базовой СК O_B
- ▶ Более подробное описание в Главе 2, [Corke(2017)]

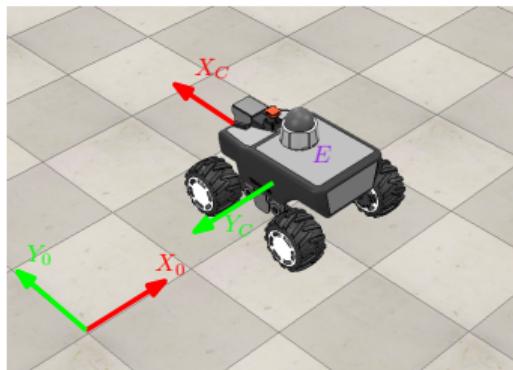


Рис. 1: Колесный робот

Положение в пространстве (3D)

- ▶ линейные координаты (x, y, z)

- ▶ угловые координаты (φ, ψ, θ)

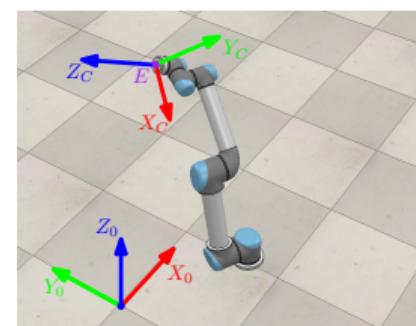


Рис. 2: Робот манипулятор

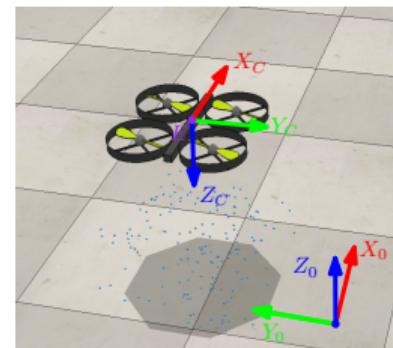
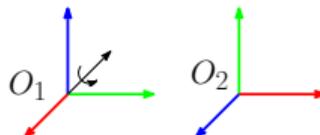
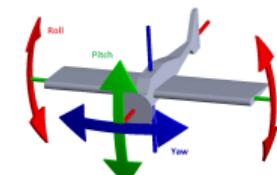
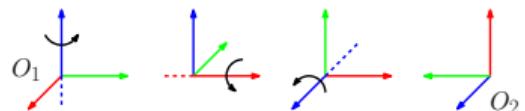


Рис. 3: Квадрокоптер

Параметризация. Задание ориентации

Наиболее используемые способы для задания ориентации

- ▶ углы Эйлера — последовательное вращение системы координат вокруг соответствующих осей (Например: XYZ , ZYZ , XYX и прочие.)
- ▶ крен-тангаж-рысканье (англ. roll-pitch-yaw) — то же, что и углы Эйлера при XYZ
- ▶ кватернион — вращение системы координат на угол θ вокруг единичного вектора \vec{n}
- ▶ Более подробно в параграфе 2.5, [Spong et al.(2006)]
- ▶ tf.transformations
 - <http://docs.ros.org/jade/api/tf/html/python/transformations.html>



Преобразования между СК

Матрица однородных преобразований, описывающая смещение и поворот СК $Ox_1y_1z_1$ относительно СК $Ox_0y_0z_0$

$${}^0H_1 = \begin{bmatrix} {}^0R_1 & {}^0\mathbf{p}_1 \\ \mathbf{0} & 1 \end{bmatrix}$$

Для преобразований на плоскости

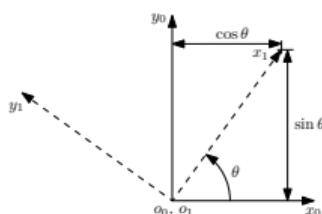
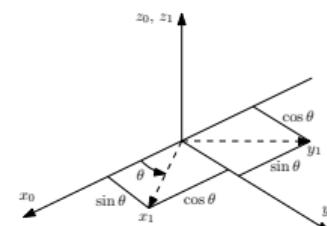


Рис. 4:

$$R = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}, \quad \mathbf{p} = \begin{bmatrix} x \\ y \end{bmatrix}, \quad (1)$$

Для преобразований в трехмерном пространстве



$$R = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad (2)$$

Преобразования между СК. ПЗК и ОЗК

- ▶ Для последовательностей СК справедливо преобразование

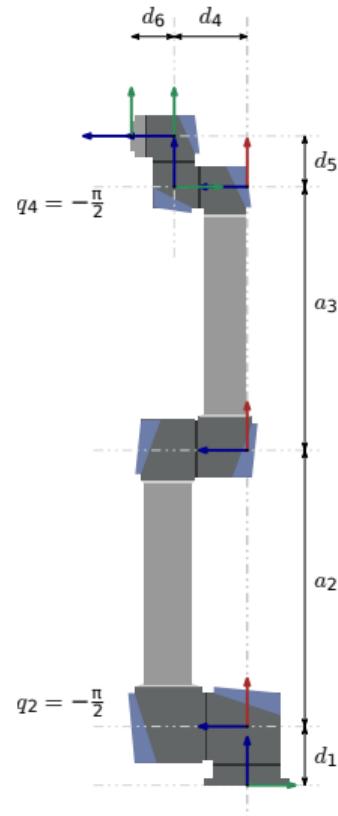
$${}^0\mathbf{r}_n = {}^0H_1{}^1H_2 \dots {}^iH_j \dots {}^{n-1}H_n \cdot {}^n\mathbf{r}_n \quad (3)$$

- ▶ Представление Денавита-Хартенберга (DH-parameters) — способ описания вращательных и поступательных связей между соседними звеньями механизма
- ▶ Прямая задача кинематики (для шестиосевого манипулятора)

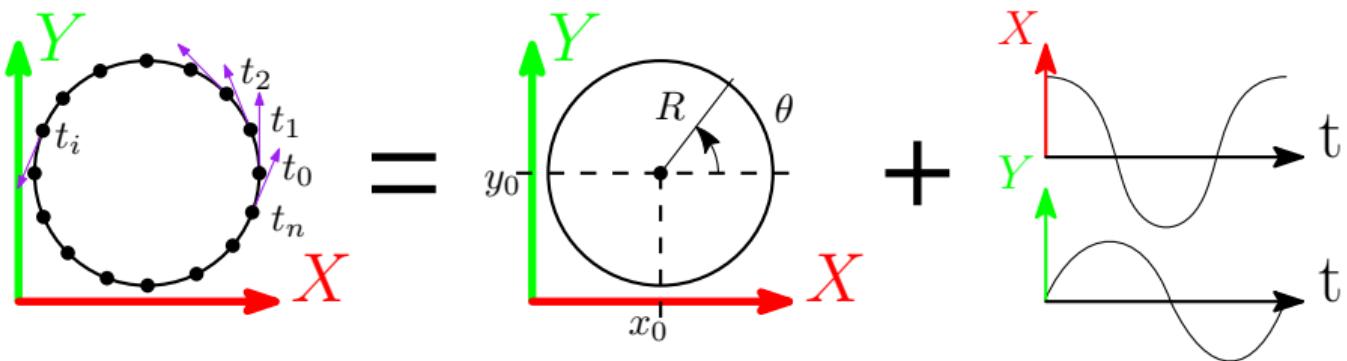
$${}^0r_6 = f(q) \quad (4)$$

- ▶ Обратная задача кинематики (для шестиосевого манипулятора)

$$q = f^{-1}({}^0r_6) \quad (5)$$

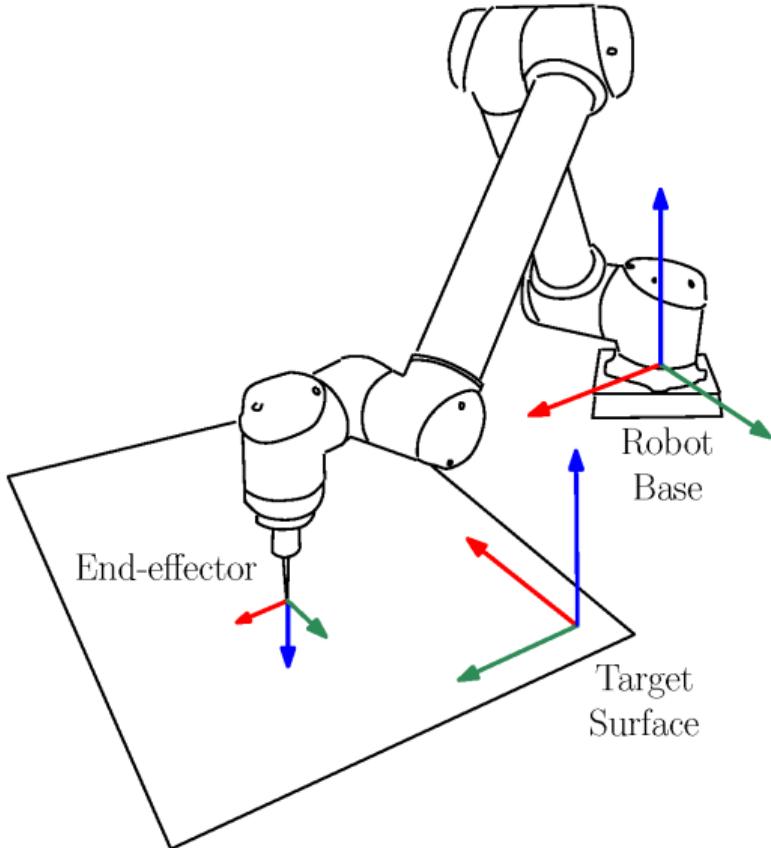


Планирование траекторий. Основы



$$l(t) = \begin{cases} x(t) = R \cos \theta(t) \\ y(t) = R \sin \theta(t) \\ \theta(t) \approx \text{atan} \left(\frac{y(t) - y(t_{prev})}{x(t) - x(t_{prev})} \right) \end{cases} \quad \begin{cases} x = R \cos \theta \\ y = R \sin \theta \\ \theta(t) = \omega t \end{cases}$$

Планирование траекторий



Планирование траекторий [Corke(2017)]

Например, для преобразования траектории из базовой СК в СК схвата

$$l_6(t) = {}^6H_0 l_0(t), \quad (6)$$

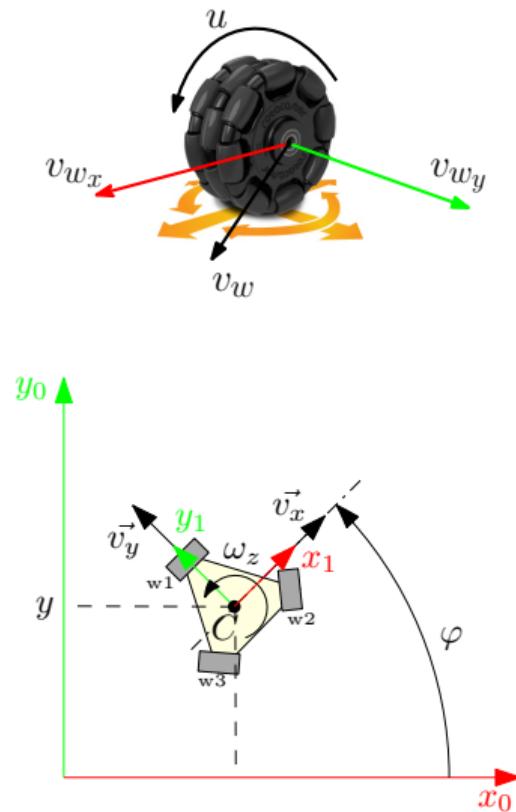
где $l_6(t)$ — траектория в СК схвата, $l_0(t)$ — траектория в СК базы.

Планирование траекторий для роботов может выполняться

- ▶ в конфигурационном пространстве
 - ▶ для манипуляторов — углы, скорости и ускорения сочленений
 - ▶ для мобильных колесных роботов — углы, скорости и ускорения колес
- ▶ в операционном пространстве
 - ▶ для манипуляторов — положения, скорости и ускорения терминальной точки
 - ▶ для мобильных роботов — положения, скорости и ускорения робота в пространстве

При планировании траектории дополнительно могут накладываться *ограничения максимальную скорость и ускорение*.

Кинематические модели (на примере колесного омнидирекционного робота)



- ▶ Кинематическая модель для робота с тремя омни колесами

$$\begin{bmatrix} \omega_z \\ v_x \\ v_y \end{bmatrix} = r \begin{bmatrix} -d & 1 & 0 \\ -d & -0.5 & -\sin(\frac{\pi}{3}) \\ -d & -0.5 & \sin(\frac{\pi}{3}) \end{bmatrix}^{-1} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \quad (7)$$

где r — радиус колес робота, d — расстояния от центра масс робота до колеса, u_i — угловая скорость i -ого колеса робота.

- ▶ Более подробно про кинематические модели роботов в Главе 13, [Lynch and Park(2017)]

Уравнение движения роботов, Глава 4, [Corke(2017)]

Динамическая модель робота в конфигурационном пространстве

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \tau, \quad (8)$$

где M, C, g — матрицы, описывающие динамику, $q = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6]^T$ — вектор обобщенных координат. Динамическая модель робота в операционном пространстве

$$\Lambda \ddot{x}_e + \mu + \rho = F_e, \quad (9)$$

где $x_e = [x, y, z, \varphi, \psi, \theta]^T$ — вектор положения.

Матрицы манипулятора в операционном пространстве

$$x_e = J_e \dot{q}, \quad \tau = J_e^T F_e$$

$$\Lambda = \left(J_e M^{-1} J_e^T \right)^{-1},$$

$$\mu = \Lambda J_e M^{-1} C(q, \dot{q}) - \Lambda j_e \dot{q}_e,$$

$$\rho = \Lambda J_e M^{-1} g(q)$$

Матрицы квадрокоптера в операционном пространстве

$$x_e = [\xi \quad \eta]^T, \quad \xi = [x \quad y \quad z]^T, \quad \eta = [\varphi \quad \psi \quad \theta]^T$$

$$\Lambda = \begin{bmatrix} m & \mathbf{0} \\ \mathbf{0} & J \end{bmatrix}, \quad \mu = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & C(\eta, \dot{\eta})\dot{\eta} \end{bmatrix}$$

$$\rho = [0 \quad 0 \quad mg \quad 0 \quad 0 \quad 0]^T$$

Управление роботами

Ошибка управления системы

$$e(t) = x(t) - x^*(t)$$

где $x(t)$ — измеренное состояние робота, $x^*(t)$ — желаемое состояние робота.

- ▶ С кинематической моделью

Пропорционально-интегрально-дифференциальный (ПИД) регулятор

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt}, \quad (10)$$

где матрицы K_p, K_i, K_d — выбираются диагональными с положительными коэффициентами на диагоналях.

- ▶ С динамической моделью

Для уравнения динамики в форме

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \tau,$$

один из наиболее простых регуляторов

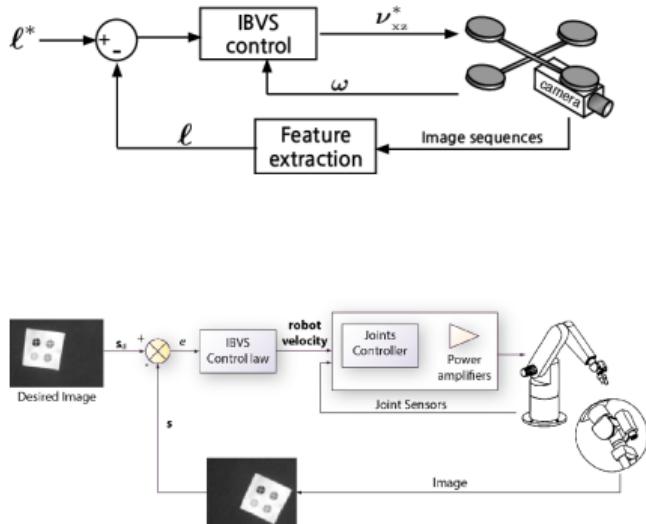
$$u_\tau(t) = K_d e(t) + K_p \frac{de(t)}{dt} + g(q).$$

- ▶ Некоторые рекомендации и методы настройки в [pid(2019a), pid(2019b)]

Управление движением роботов. Visual servoing. Концепция

Основаная идея — использование данных с визуальных сенсоров в обратной связи системы управления.

Основные методы



- ▶ Image-based (IBVS) — в управлении участвует непосредственно информации из изображения ("в пикселях")
- ▶ Position/pose-based (PBVS) — после извлечения информации из изображения, производится оценка положения объекта, например, относительно СК камеры
- ▶ Гибридный подход — различные модификации, использующие положительные стороны предыдущих методов

Управление движением роботов. Visual servoing

Некоторые замечания, относительно управления квадрокоптером
 (Часть 5, [Corke(2017)])

- ▶ Квадрокоптер управляет вектором скорости $\dot{q}_e = [v_x, v_y, v_z, \omega_z]^T$
- ▶ Пространственная скорость квадракоптера
 - ▶ в СК квадракоптера $v_e = [v_x, v_y, v_z, \omega_x, \omega_y, \omega_z]^T$
 - ▶ в СК камеры $v_c = [v_x, v_y, v_z, \omega_x, \omega_y, \omega_z]^T$
- ▶ ${}^c M_e$ — матрица преобразования между системами координат квадракоптера и камеры
- ▶ используемые признаки (features) $s = [n_{g_x}, n_{g_y}, n_\alpha, \text{atan} \frac{1}{\rho}]^T$
 - ▶ $n_g = [n_{g_x}, n_{g_y}]$ положение маркера в пространстве изображения камеры ($O_{X_c Y_c}$)
 - ▶ n_α — матрица, зависящая от площади маркера
 - ▶ ρ — радиус из пары полярных координат (ρ, θ)

Контроллер имеет вид

$$\dot{q}_e = -\lambda(L_s^c V_e^e J_e)^+(s - s^*), \quad (11)$$

где λ — положительное число, ${}^c V_e$ — матрица преобразования скоростей $v_c = {}^c V_e v_e$, ${}^e J_e$ — матрица Якоби квадрокоптера для пересчета вектора скорости задания $v_e = {}^e J_e \dot{q}_e$.

Базовые алгоритмы технического зрения



Основные моменты

- ▶ Цвет, Глава 10, [[Corke\(2017\)](#)]
- ▶ Представление изображений, Глава 11, [[Corke\(2017\)](#)]
- ▶ Обработка изображений (фильтрация [[cvf\(2019b\)](#)], выделение границ [[cve\(2019\)](#)], поиск контуров [[cvc\(2019\)](#)], сегментация [[cvs\(2019\)](#)]), Главы 12-13, [[Corke\(2017\)](#)])
- ▶ Извлечение признаков (англ. feature) [[cvf\(2019a\)](#)]

Прочие направления в компьютерном зрении

- ▶ Стереоизображения (эпиполярная геометрия)
- ▶ RGB-D изображения

Кик-старт по ПО

Ubuntu

- ▶ Создание загрузочной USB флешки [[usb\(2019\)](#)]
- ▶ Установка Linux Ubuntu
 - ▶ как основной системы [[mai\(2019\)](#)]
 - ▶ как второй системы (dual boot) [[dua\(2019\)](#)]
 - ▶ на виртуальную машину (virtualbox) [[vbo\(2019\)](#)]
- ▶ Введение в Linux Ubuntu и терминал [[ter\(2019\)](#)]

Внимание!

Установка операционной системы — ответственная и сложная процедура.

Проводите ее только, если в полной мере понимаете что делаете.

Кик-старт по ПО

Знакомство с docker

Знакомство с docker

- ▶ Концепция [con(2019)]
- ▶ Терминология и как использовать [how(2019)]

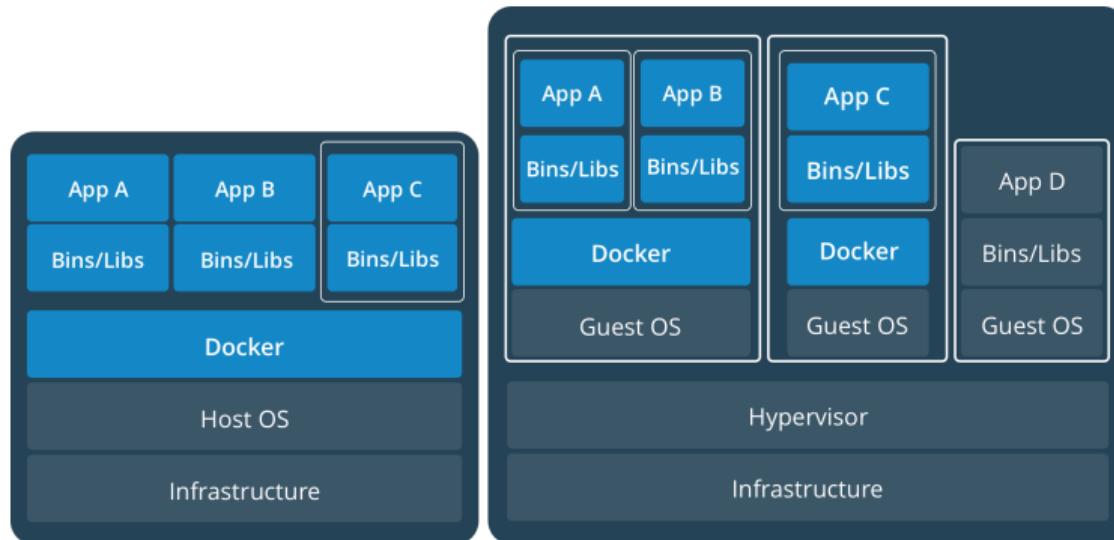


Рис. 5: Отличие docker от виртуальной машины

Кик-старт по ПО

Установка и настройка docker

Установка и настройка ПО для решения задач на олимпиаде (для пользователей Linux)

- ▶ Установите систему контроля версии git

```
$ sudo apt-get install git
```

- ▶ Склонируйте репозиторий с ПО

```
$ cd /home/$USER
```

```
$ git clone https://github.com/be2rlab/yaprofi\_robotics\_2020.git
```

- ▶ Определите тип видео карты, выполнив в терминале

```
$ sudo lshw -C video
```

```
product: GP107 [GeForce GTX 1050 Ti]
```

- ▶ Запустите скрипт установки и настройки docker

```
$ cd /home/$USER/yaprofi_robotics_2020/docker/
```

```
$ sudo "производитель_видео_карты"/install_docker.bash
```

- ▶ Запустите скрипт сборки docker контейнера

```
$ sudo "производитель_видео_карты"/build_docker.bash
```

Кик-старт по ПО

Проверка и использование docker контейнера

Проверка, что ПО работает (в частности, что работает 3D)

- ▶ Перейти в

```
$ cd /home/$USER/yaprofi_robots_2020/docker/
```

- ▶ Запустить тестовый скрипт

```
$ sudo "производитель_видео_карты"/run_docker.bash
```

- ▶ Далее внутри контейнера

```
# cd /opt/csim
```

```
# ./coppeliaSim.sh
```

Кик-старт по ПО

Проверка и использование docker контейнера

Запуск симулятора CoppeliaSim из docker контейнера

▶ Перейти в

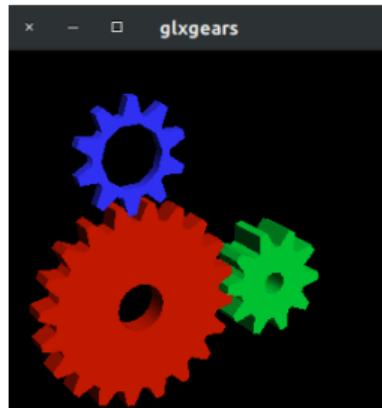
```
$ cd /home/$USER/yaprofi_robots_2020/docker/
```

▶ Запустить скрипт

```
$ sudo "video_driver"/test_gui_docker.bash
```

Замечание

Все настроено правильно, если вы видите анимацию "шестерней" и вывод в терминале



```
root@bird:/# glxgears
Running synchronized to the vertical refresh. The framerate should be
approximately the same as the monitor refresh rate.
300 frames in 5.0 seconds = 59.960 FPS
301 frames in 5.0 seconds = 60.003 FPS
301 frames in 5.0 seconds = 60.003 FPS
301 frames in 5.0 seconds = 60.004 FPS
^C
root@bird:/#
```

Кик-старт по ПО

Знакомство с ROS

ROS



- ▶ *Robot Operation System (ROS)* — коллекция библиотек и инструментов, упрощающих разработку ПО для роботов
- ▶ Основные языки программирования *C++* и *Python2*
- ▶ Все дистрибутивы *ROS* можно найти в [\[ros\(2019a\)\]](#)
- ▶ Первые шаги в *ROS* можно начать с уроков в [\[ros\(2019b\)\]](#)

Кик-старт по ПО

Философия ROS

Концепции ROS

- ▶ Master
- ▶ Nodes
- ▶ Topics
- ▶ Message

- ▶ Master — это основаная программа в ROS, которая соединяет между собой ноды (программы написанные пользователями и из библиотек)
 - ▶ запускается только один раз и работает на протяжении всей сессии
 - ▶ при перезапуске, все ноды, также, перезапускаются
- ▶ Команда для запуска Мастера
`$ roscore`

Кик-старт по ПО

Философия ROS

Концепции ROS

- ▶ Master
- ▶ **Nodes**
- ▶ **Topics**
- ▶ Message

- ▶ Node (Нода) — это программы с уникальным именем.
Обычно реализуется алгоритмы, логику и проч.
 - ▶ Ноды могут связываться друг с другом через топики (topics)
 - ▶ Любая нода может принимать или пересыпать данные любой ноде
 - ▶ Нода, которая отправляет данные в заданный топик, называется Publisher
 - ▶ Нода, которая принимает данные, называется Subscriber
 - ▶ Данные, которые пересыпаются между нодами через топики, называются Message
- ▶ Для запуска ноды
`$ rosrun package_name node_name`
- ▶ Для визуализации архитектуры приложения (или всех запущенных сейчас нод) с информацией о связях, нужно вызвать команду
`$ rqt_graph`

Кик-старт по ПО

Философия ROS

Концепции ROS

- ▶ Master
- ▶ **Nodes**
- ▶ **Topics**
- ▶ Message

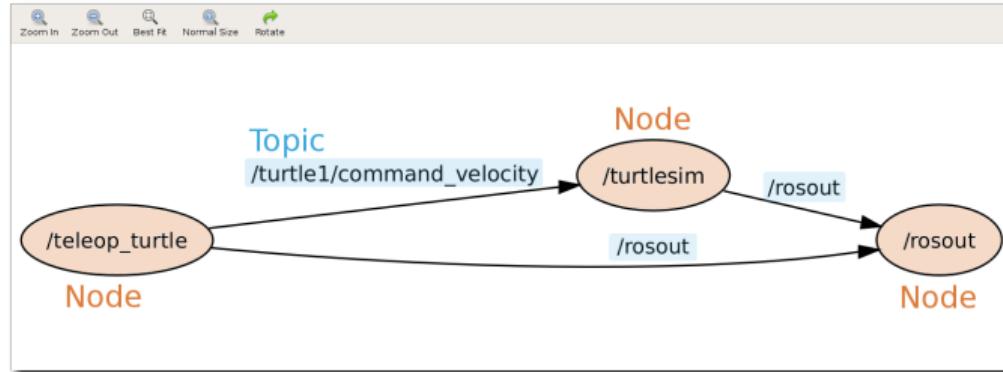


Рис. 6: Пример вывода нод и топиков в rqt_graph

Кик-старт по ПО

Философия ROS

Концепции ROS

- ▶ Master
- ▶ Nodes
- ▶ Topics
- ▶ **Message**

- ▶ Каждое сообщение (Message) имеет собственный тип
- ▶ Тип сообщения в любом из топиков возможно определить
`$ rostopic type topic_name`
- ▶ Тип сообщения (Message) состоит из двух частей
`std_msgs/Int8`
 - ▶ `std_msgs` — название пакета, где расположен исходный код
 - ▶ `Int8` — название типа сообщения

Кик-старт по ПО

Симулятор CoppeliaSim (бывш. V-REP)



**COPPELIA
ROBOTICS**

CoppeliaSim это симулятор робототехнических систем. Управление осуществляется либо при помощи встроенных скриптов на языке Lua, либо с использованием интерфейсов (плагинов) к различным фреймворкам: ROSInterface, remoteAPI и прочие, что позволяет программировать робототехнические системы и их контроллеры на C/C++, Python, Java, Lua, Matlab, Octave or Urbi. Для более близкого знакомства следует обратиться по следующим ссылкам

- ▶ Руководство пользователя с официального сайта [[csi\(2019\)](#)]
- ▶ Знакомство с интерфейсом и функционалом на [habr.com](#) [[vre\(2019a\)](#), [vre\(2019b\)](#)]

Кик-старт по ПО

Симулятор CoppeliaSim (бывш. V-REP)

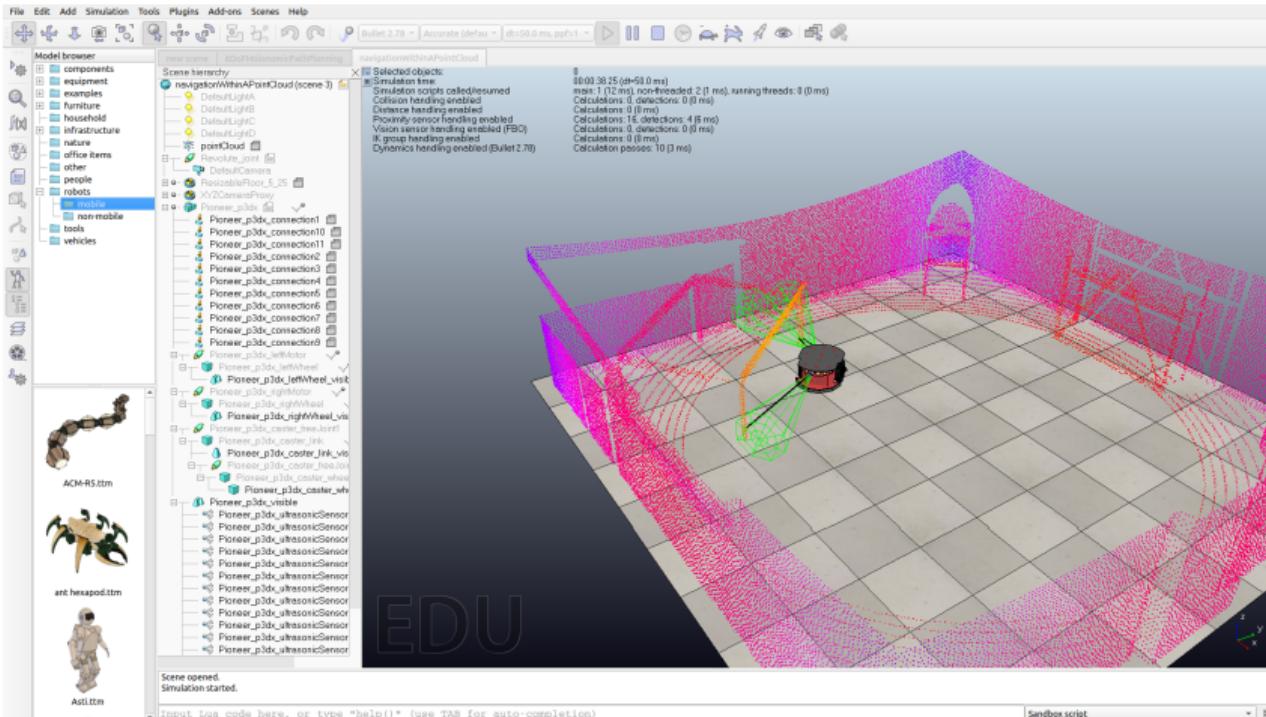


Рис. 7: Интерфейс симулятора CoppeliaSim

Кик-старт по ПО

Доступные для использования библиотеки

Ссылки на документацию для ознакомления с функционалом

- ▶ ROS [[ros\(2019b\)](#)]
- ▶ CoppeliaSim [[csi\(2019\)](#)]
- ▶ OpenCV 3.2 — библиотека алгоритмов компьютерного зрения [[ope\(2019\)](#)]
- ▶ VISP — библиотека алгоритмов управления с визуальной обратной связью [[vis\(2019\)](#)]
- ▶ eigen3 — C++ библиотека для работы с линейной алгеброй [[eig\(2019\)](#)]
- ▶ numpy, scipy — библиотеки для работы с математическими структурами в Python

Bibliography I

-  Docker concepts.
<https://tproger.ru/translations/docker-concepts/>, 2019.
[Online; accessed 23-Января-2020].
-  CoppeliaSim Tutorials.
<http://www.coppeliarobotics.com/helpFiles/>, 2019.
[Online; accessed 23-Января-2020].
-  OpenCV, contours.
https://docs.opencv.org/master/df/d0d/tutorial_find_contours.html,
2019.
[Online; accessed 23-Января-2020].

Bibliography II

-  OpenCV, Edges.
https://docs.opencv.org/master/da/d5c/tutorial_canny_detector.html, 2019.
[Online; accessed 23-Января-2020].
-  OpenCV, Features.
https://docs.opencv.org/master/d9/d97/tutorial_table_of_content_features2d.html, 2019a.
[Online; accessed 23-Января-2020].
-  OpenCV, Filtering.
https://docs.opencv.org/master/dc/dd3/tutorial_gaussian_median_blur_bilateral_filter.html, 2019b.
[Online; accessed 23-Января-2020].

Bibliography III

-  OpenCV, Distance transform.
https://docs.opencv.org/master/d2/dbd/tutorial_distance_transform.html, 2019.
[Online; accessed 23-Января-2020].
-  Dualboot Ubuntu intstall.
<https://lifehacker.ru/kak-ustanovit-linux>, 2019.
[Online; accessed 23-Января-2020].
-  Eigen.
<http://eigen.tuxfamily.org/>, 2019.
[Online; accessed 23-Января-2020].

Bibliography IV

-  How to start using docker.
<https://tproger.ru/translations/how-to-start-using-docker/>, 2019.
[Online; accessed 23-Января-2020].
-  Mainboot Ubuntu install.
<https://tutorials.ubuntu.com/tutorial/tutorial-install-ubuntu-desktop>,
2019.
[Online; accessed 23-Января-2020].
-  OpenCV.
https://docs.opencv.org/master/d9/df8/tutorial_root.html, 2019.
[Online; accessed 23-Января-2020].

Bibliography V

-  Расчет параметров ПИД регулятора.
https://www.bookasutp.ru/Chapter5_5.aspx, 2019a.
[Online; accessed 23-Января-2020].
-  What are good strategies for tuning PID loops?
<https://robotics.stackexchange.com/questions/167/what-are-good-strategies-for-tuning-pid-loops>, 2019b.
[Online; accessed 23-Января-2020].
-  ROS distributions.
<http://wiki.ros.org/Distributions>, 2019a.
[Online; accessed 23-Января-2020].
-  ROS tutorials.
<http://wiki.ros.org/ROS/Tutorials>, 2019b.
[Online; accessed 23-Января-2020].

Bibliography VI

-  Command line for beginners.
<https://tutorials.ubuntu.com/tutorial/command-line-for-beginners>, 2019.
[Online; accessed 23-Января-2020].
-  Create usb flash drive, Ubuntu.
<https://tutorials.ubuntu.com/tutorial/tutorial-create-a-usb-stick-on-windows>, 2019.
[Online; accessed 23-Января-2020].
-  Install ubuntu on VirtualBox.
<https://www.wikihow.com/Install-Ubuntu-on-VirtualBox>, 2019.
[Online; accessed 23-Января-2020].

Bibliography VII

-  VISp.
<https://visp-doc.inria.fr/doxygen/visp-3.2.0/>, 2019.
[Online; accessed 23-Января-2020].
-  V-REP — гибкая и масштабируемая платформа для робомоделирования.
Часть 1.
<https://habr.com/ru/post/383009/>, 2019a.
[Online; accessed 23-Января-2020].
-  V-REP — гибкая и масштабируемая платформа для робомоделирования.
Часть 2.
<https://habr.com/en/post/385725/>, 2019b.
[Online; accessed 23-Января-2020].

Bibliography VIII



Peter Corke.

Robotics, vision and control: fundamental algorithms in MATLAB® second, completely revised, volume 118.

Springer, 2017.



Kevin M Lynch and Frank C Park.

Modern Robotics.

Cambridge University Press, 2017.



Mark W Spong et al.

Robot modeling and control.

2006.