Интеллектуальные робототехнические системы

2022/23 учебный год

Второй отборочный этап

Задачи второго тура решают один из основных вызовов робототехники — автономная навигация мобильных платформ. Похожие задания можно встретить в таких международных соревнованиях как: RoboCup (https://www.robocup.org/leagues/16), Robotex Internation (https://robotex.international/), Darpa Robotic Challange (https://www.darpa.mil/program/darpa-robotics-challenge). Мы постарались приблизить задачи к реальности, их развитие будет представлено в финале уже на реальных роботах. Аналогичные задачи решаются по отношению к любому автопилоту на автономных складах, в роботах пылесосах и системе доставки.

Каждый член команды должен взять на себя одну из четырех ролей:

1. **Математик-алгоритмист** — подбор и разработка алгоритмов для программистов: управления и навигации, создания графа занятости, алгоритмов поиска пути в графе, построения траектории и т. д.
2. **Программист по управлению роботом C++/Python** — программирование системы управления, функционирующей на роботе, для решения задач коммуникации, движения, навигации, работы с датчиками, одометрии и т. д.
3. **Программист компьютерного зрения и высокоуровневых систем C++/Python** — программирование высокоуровневой системы управления (распределение задач, поиск пути и т. д.), обработки изображения, решения задач коммуникации с системой управления роботом.
4. **Капитан команды** — распределение задач, расстановка приоритетов, распределение ресурсов, разрешение споров.

Участники должны знать основы математического анализа, алгебры, геометрии, основы программирования на C/C++ и Python. Участникам требуются знания алгоритмов поиска пути, планирования движения, алгоритмов управления, алгоритмов обработки изображения, знание библиотеки OpenCV.

Ниже представлены Задания на программирование мобильной платформы. Вам предоставляется симулятор, адаптированный под каждую задачу. Описать решение задачи необходимо в файле `task1.cpp`, `task2.cpp`, ... или `task1.py`, `task2.py`, ... в зависимости от выбранного языка программирования и номера задачи. Файлы уже подготовлены для работы, ваша задача написать программный код в функцию `solve`. В каждой задаче функция принимает объект симулятора для управления роботом и дополнительные данные для выполнения Задания. Будьте внимательны к требованиям к входным и выходным значениям функции. Допускается написание дополнительных классов, функций и импорт дополнительных библиотек внутри файла **с решением** `task*`, однако их набор ограничен в силу ограниченности проверяющей системы (список доступных библиотек представлен в приложении). Инструкция по скачиванию и установке также описана далее.

Инструкцию по скачиванию, установке и запуску симулятора для начала выполнения задания можно найти по ссылке: https://gitlab.com/ovcharov.alex.o/nto_robotics. Здесь же лежат папки с задачами, где подготовлены файлы `task*.py` и `task*.cpp`. Там также можно найти описание функции solve и минимально рабочий пример работы с системой для быстрого старта группы участников.

Задача IV.1. Движение по восьмерке (20 баллов)

Темы: мобильная робототехника, дифференциальная платформа, одометрия, траекторное управление, слежение за траекторией, компьютерное зрение.

Условие

В задаче транспортировки объектов мобильные существует подзадача, где роботы строят глобальный маршрут исходя из карты местности. Робот должен придерживаться линии, чтобы избежать столкновений и достичь поставленной цели, здесь начинает работать траекторный регулятор. Он выполняет функцию слежения за целевой (той, что задал планировщик маршрута) траекторией, т. е. минимизирует разницу между положением робота и траекторией. В задаче ниже мы предоставим вам 2D симулятор с полем, где в центре расположена линия. Она моделирует маршрут, который теоретически может построить планировщик с учетом стен, столов и других препятствий. Вы должны проехать по этой линии несколько кругов и отправится в зону окончания задания.

Введем основные обозначения, которыми будем пользоваться далее в условиях задач.

Участнику предоставляется симулятор, где в программе публикуется следующая информация:

1. Угол левого и правого колеса α_l, α_r, мобильной платформы.
2. Угловая скорость левого и правого колеса $\dot{\alpha}_l$, $\dot{\alpha}_r$, мобильной платформы.
3. Изображение комнаты с мобильной платформой.
4. Изображение комнаты с основными обозначениями (названием зон).

Участник может управлять мобильной платформой, задавая напряжение на левом и правом двигателе.

Про изображение комнаты. Мы постарались смоделировать план помещения и представить, что на него постоянно сверху смотрит 2D камера. На изображении присутствуют:

- стены (черные);
- маркеры (целевая линия движения);
- мобильная платформа (робот);
- зоны (отмечены красным цветом):
  - точка старта на линии;
  - зона старта (`START`) — прямоугольник с фиксированными размерами;
  - зона окончания задания (`EXIT`) — прямоугольник с фиксированными размерами.

Участнику предоставляется изображение помещения со всеми обозначениями и пример того, что участник увидит в симуляторе. На изображении представлена целевая линия (серая линия), по которой должен следовать робот. Толщина линии везде одинаковая и лежит в пределах от 10 до 50% от ширины робота. Толщина линии может отличатся в судейских Тестах (смотри задание) от тех тестов, что открыты участникам.

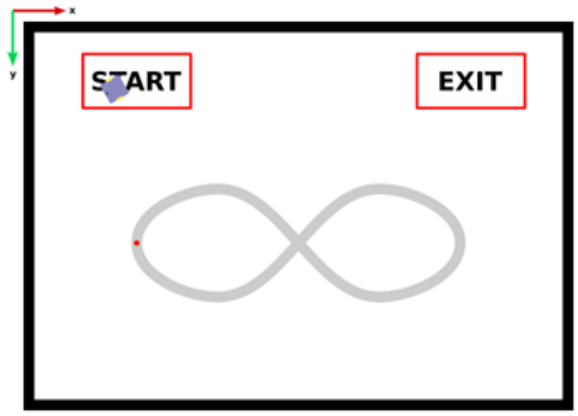


Рис. 1. Карта помещения с основными обозначениями

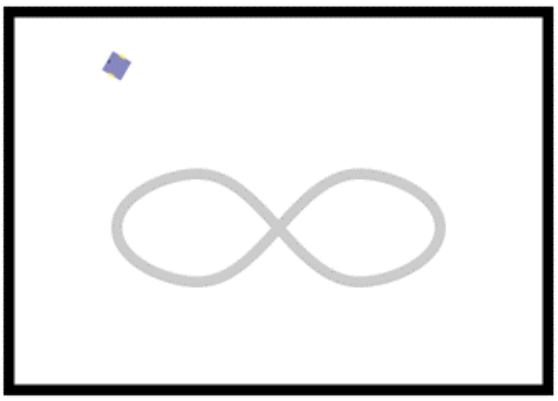


Рис. 2. Данные, которые видит участник в симуляторе

Изображение помещения с мобильной платформой внутри

В симуляторе карта обновляется с заданной частотой. Оси карты направлены также, как и в библиотеке OpenCV (x — вправо, y — вниз), точка отсчета находится в левом верхнем углу картинки.

Про мобильную платформу. Она представляет собой дифференциальную платформу с двумя колесами и располагается на стартовой позиции в помещении. Робот представлен на рисунке 3, слева представлена графическая модель, чей рендер отрисован на изображении помещения. Размер робота во всех тестах одинаковый, ширина примерно равна высоте.

Ниже, слева направо вводятся основные понятия и точки на роботе. Ориентация робота θ считается от горизонтальной оси x, положительное направление вращения робота совпадает с направлением движения стрелки часов (по часовой стрелке). Положение робота считается от левого верхнего угла изображения помещения.

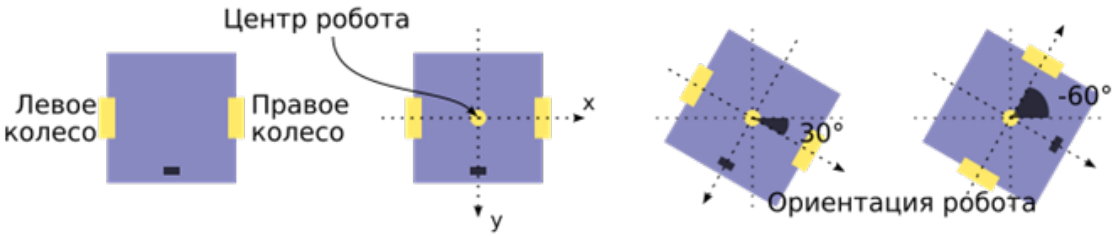
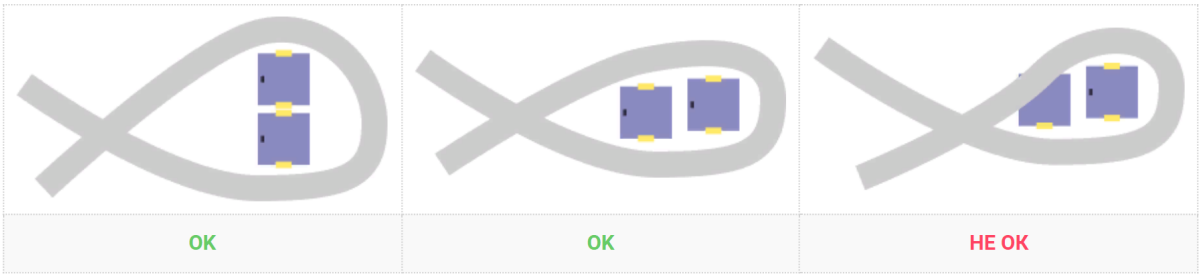


Рис. 3. Графическая модель робота и основные обозначения

Задание. После запуска симулятора робот расположен в зоне `START` (смотри рисунок 1–2) со случайной ориентацией. Задача робота проехать n кругов и заехать в зону `EXIT`. Задание разделено на несколько тестов, их количество будет объявлено позже. В симуляторе вам предоставлен к решению только первый тест, есть также три судейских теста с другими линиями. Ваше решение будет отправлено на сервер и пройдет через три дополнительных судейских теста. Линия в помещении обязательно замкнутая и имеет одну петлю, куда могут поместиться как минимум два робота (смотри рисунок для примера), это сделано, чтобы ограничить радиус поворота петли, она не может поворачивать слишком резко. Это должно пригодиться при работе с регулятором.



Помимо восьмерки линия может также иметь множество поворотов, в том числе и крутых.

В случае успешного завершения теста вы получаете полный балл. Предусмотрено **начисление** баллов движение по линии и начало движения по ней, снятие баллов за съезд с линии и до полной **отмены** теста (участник **получает 0 баллов**). Далее подробней рассмотрим задание и судейство для лучшего понимания.

1. Сразу после старта симуляции запускается таймер. Время выполнения теста ограничено.
2. После запуска робота он должен рассчитать траекторию и начать движение к красной точки старта на линии.
3. Считается, что начал движение по линии тогда, когда его центр оказался в пределах красной точки старта на линии (окружности диаметром в ширину линии).
4. После начала движения по линии выезжать за ее пределы нельзя. Если центр робота выезжает за пределы линии задание заканчивается (прерывание теста), участник **получает 0 баллов** за тест.
5. Регламентируется лишь то, что робот должен двигаться вдоль линии в одном и том же направлении, что центр робота должен оставаться на линии и правила проезда перекрестия на линии.
6. Мы разделили линию на окружности и проверяем, чтобы участник проходил из одной в другую. Считается, что робот выполнил круг лишь тогда, когда его центр попал в окрестность красной точки (окружность, из которой началось движение по линии).

Рис. 4. Разделение линии на окружности

7. Направление движения по линии **фиксируется** роботом сразу как он начнет движение вдоль нее и далее **не меняется**. Это значит, что робот заехал в стартовую окружность и заехал в соседнюю, так вычисляется направление движения.

8. При проезде перекрестия восьмерки нужно сохранять траекторию движения «прямо по линии». Отклонение и поворот влево или право будет оцениваться в 0 баллов с прерыванием теста.

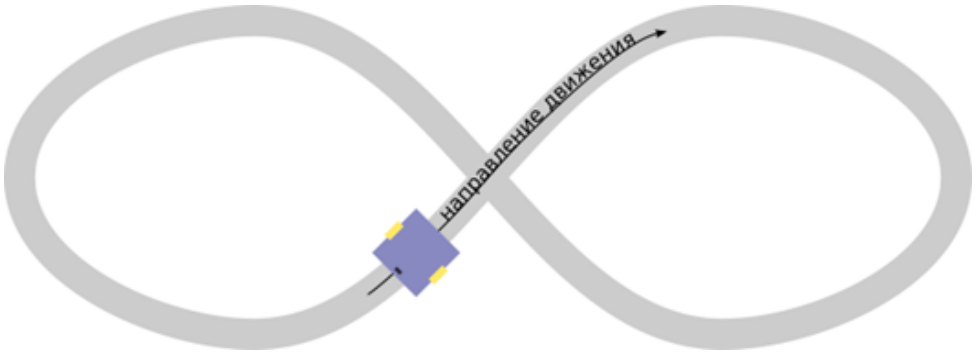


Рис. 5. Направление движения через перекрестие

9. После завершения движения по линии (робот должен проехать n кругов), робот должен поехать в зону `EXIT`. Сразу, как центр робота достигнет зоны `EXIT` задание считается выполненным.
10. Тест прерывается и участник **получает 0 баллов** при наезде на **черные стены**. Робот не должен пересекать стены. Робот может двигаться только по белым частям помещения и серым маркерам.
11. Тест прерывается и участник **получает 0 баллов**, если у участника не осталось времени на выполнение теста.

Участник должен написать решение в функции `solve`. В данном задании в качестве аргумента функции поступает объект симулятора, который позволяет управлять роботом и дает всю необходимую информацию о роботе (параметры кинематической модели, углы и скорости колес) и о карте (рендр помещения с роботами, изображение помещения с обозначениями).

Инструкцию по скачиванию, установке и запуску симулятора для начала выполнения задания можно найти по ссылке: https://gitlab.com/ovcharov.alex.o/nto_robotics. Здесь же лежат папки задачами, где подготовлены файлы `task*.py` и `task*.cpp`. Там также можно найти описание функции `solve` и минимально рабочий пример работы с системой для быстрого старта группы участников.

Решение

Решение может быть разным, допускается любое, удовлетворяющее правилам. Наиболее простым является:

1. Обрабатывая изображение комнаты, определяем координаты точек в центре кривой, заполняем ими массив. В качестве первого элемента выбираем координаты красной точки.
2. Обрабатывая изображение комнаты, определяем координаты центра робота.
3. Выбираем красную точку как целевую в движении к линии.
4. Строим прямой путь до целевой точки, разбиваем его на множество точек и определяем желаемую ориентацию робота (робот должен быть ориентирован по прямой).

5. Разворачиваем робота к желаемой ориентации с помощью ПИД регулятора ориентацией.
6. Движемся к желаемой точке на кривой. Для этого:
   6.1. в цикле передаем каждый раз следующую точку на прямой;
   6.2. обрабатывая изображение комнаты, координаты центра робота;
   6.3. считаем скорость робота через кинематическую модель и известную угловую скорость;
   6.4. на основе центра робота и точки на кривой формируем вектор желаемой скорости и отправляем вместе с рассчитанной скоростью на ПИД регулятор скоростью робота;
   6.5. движемся так до конца линии (до последней точки в массиве).
7. Аналогично пункту 6, движемся вдоль кривой линии по точкам из пункта 1. Считаем вектор скорости и отправляем на ПИД регулятор скоростью робота. Повторяем так n кругов.
8. Обрабатывая изображение комнаты с обозначениями, определяем точку в зоне `EXIT` аналогично пунктам 1, 2, 3, 4, 5 движемся в зону.

Ссылка на тесты с симулятором и инструкцией по скачиванию, установке и запуску: https://gitlab.com/ovcharov.alex.o/nto_robotics.

Задача IV.2. Движение по маркерам в свободном пространстве (25 баллов)

Темы: мобильная робототехника, дифференциальная платформа, одометрия, методы планирования траектории, компьютерное зрение.

Условие

Введем основные обозначения, которыми будем пользоваться далее в условиях задач.

Участнику предоставляется симулятор, где в программе публикуется следующая информация:

1. Угол левого и правого колеса α_l, α_r мобильной платформы.
2. Угловая скорость левого и правого колеса $\dot{\alpha}_l$, $\dot{\alpha}_r$ мобильной платформы.
3. Изображение комнаты с мобильной платформой.
4. Изображение комнаты с основными обозначениями (названиями маркеров и зон).

Участник может управлять мобильной платформой, задавая напряжение на левом и правом двигателе.

Про изображение комнаты. Мы постарались смоделировать план помещения и представить, что на него постоянно сверху смотрит 2D камера. На изображении присутствуют:

- стены (черные);
- маркеры (целевая линия движения);
- мобильная платформа (робот);
- зона старта (`START`);

- зона окончания задания (`EXIT`).

Маркеры (`P1`, `P2`, `P3`, `P4` и т. д.) означают возможную зону (квадрат), в которую нужно будет приехать. Участнику предоставляется изображение помещения со всеми обозначениями и пример того, что участник увидит в симуляторе. Размер, цвет и форма маркеров во всех тестах одинаковые. На изображении комнаты с основными обозначениями присутствуют имена маркеров (`P1`, `P2` ...), они представлены только для примера. Участнику предстоит **самостоятельно присвоить имя маркеру**, на изображении представлен лишь пример того, как это может выглядеть.

Рис. 6. Карта помещения с основными обозначениями

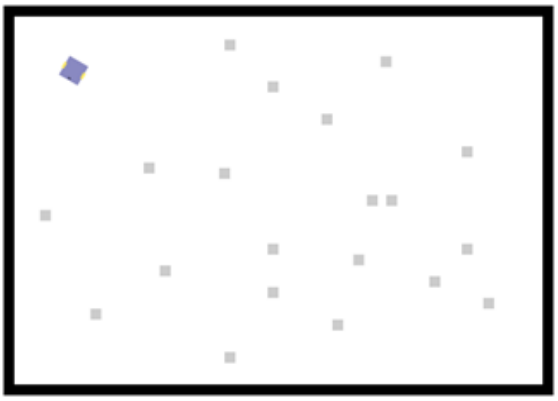
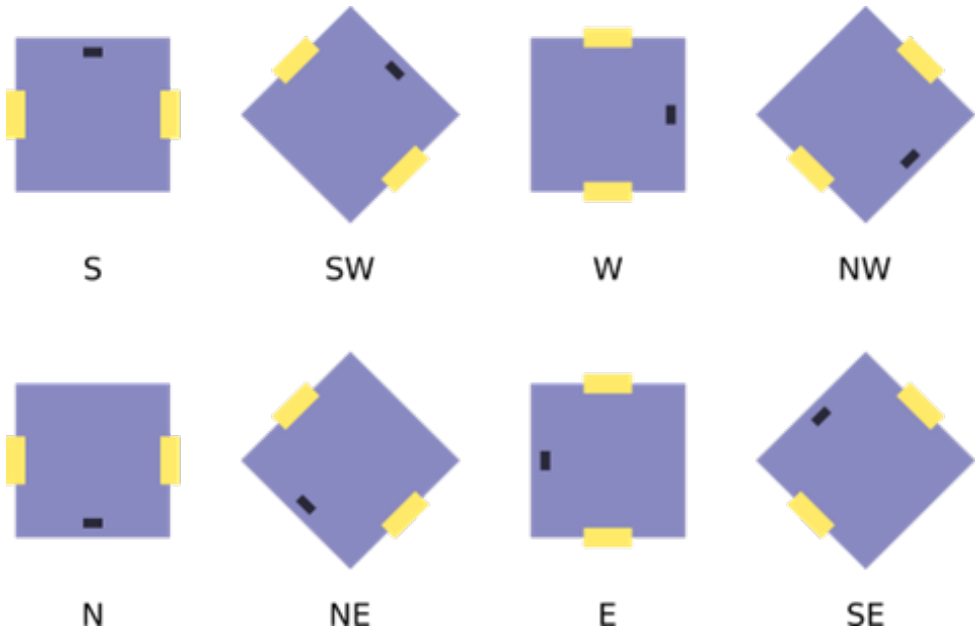


Рис. 7. Данные, которые видит участник в симуляторе

Изображение помещения с мобильной платформой внутри

В симуляторе карта обновляется с заданной частотой (**смотри параметры к задаче на следующем шаге**). Оси карты направлены также, как и в библиотеке OpenCV (x — вправо, y — вниз), точка отсчета находится в левом верхнем углу картинки. Также выделено восемь направлений: юг (S), север (N), запад (W), восток (E), юго-восток (SE), юго-запад (SW), северо-восток (NE), северо-запад (NW), они наглядно представлены на рисунке ниже.



Про мобильную платформу. Она представляет собой дифференциальную платформу с двумя колесами и располагается на стартовой позиции в помещении. Робот представлен на рисунке 8, слева представлена графическая модель робота, чей рендер отрисован на изображении помещения. Размер робота во всех Тестах одинаковый, ширина примерно равна высоте.

Ниже, слева направо вводятся основные понятия и точки на роботе. Ориентация робота θ считается от горизонтальной оси x, положительное направление вращения робота совпадает с направлением движения стрелки часов (по часовой стрелке). Положение робота считается от левого верхнего угла изображения помещения.

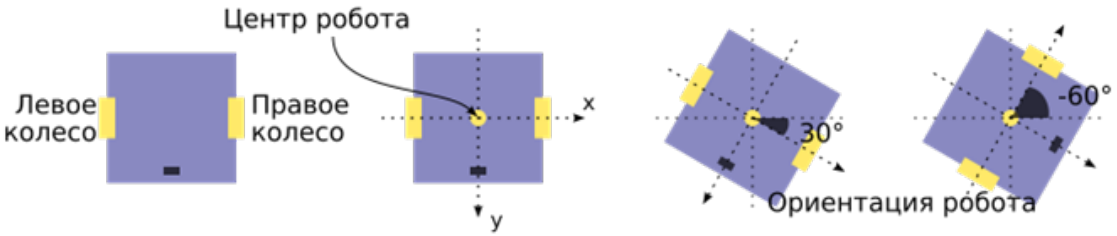


Рис. 8. Графическая модель робота и основные обозначения

Задание. После запуска симулятора робот расположен в зоне `START` (смотри рисунок 6–7) со случайной ориентацией. Задание разделено на несколько **карт**. В симуляторе вам предоставлена к решению только **первая карта**, есть также три судейских карты с иным расположением маркеров (форма и размер не меняются).

Для каждой **карты** студент должен распознать и передать количество, имя маркеров и их координаты на изображении по шаблону «`P{№ маркера} {координата x} {координата y}`» (без кавычек). Пример:

```
5
P1 235 328
P2 15 92
P3 34 589
p4 858 322
P5 134 324
```

Важно сохранить далее эти имена, т. к. система будет использовать их для проверки. Для каждой **карты** генерируются **тесты**, в которых система создает строку со списком из N названий маркеров (имен, что участник передал ранее). Робот должен посетить каждый из этих маркеров. Участнику поступят следующие значения:

1. на первой строке указано количество меток, которые робот должен посетить,
2. номер метки и ориентация, с которой робот должен прийти в заданную метку.

Пример строки с заданием.

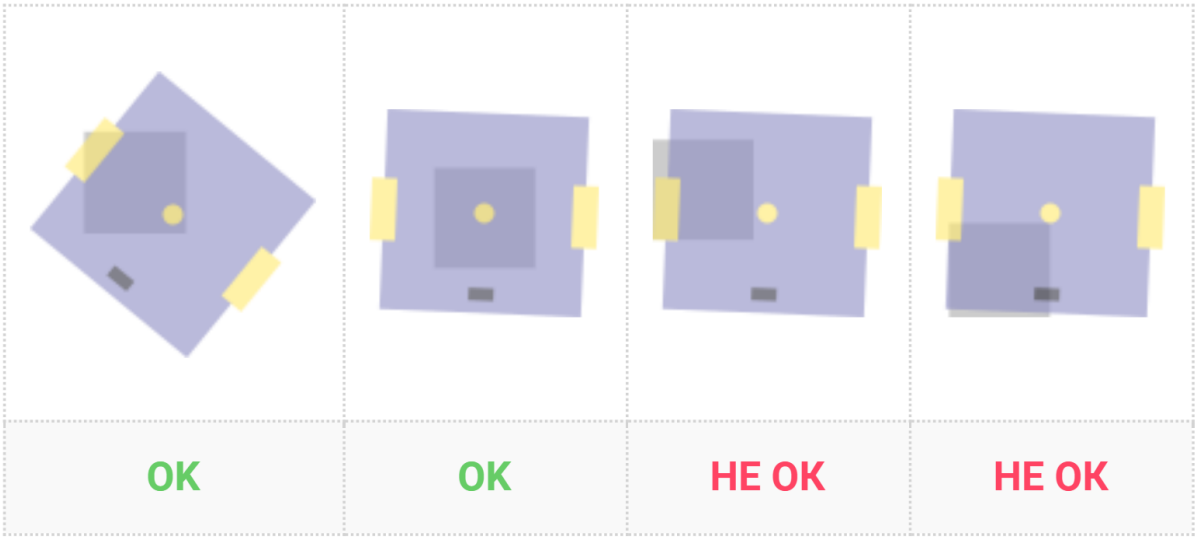
```
9
P3_SW P2_S P4_W P5_NE P16_W P8 P13_N P15 P16
```

В случае успешного завершения теста вы получаете полный балл. Предусмотрено начисление баллов за достижение маркеров (робот заехал на маркер с заданной ориентацией).

1. Сразу после старта симуляции запускается таймер. Время выполнения теста ограничено.
2. После запуска робота он должен рассчитать траекторию и начать движение к первому маркеру.
3. После первого маркера из списка робот должен рассчитать траекторию и начать движение ко второму маркеру из списка и так далее.
4. Важно сохранять последовательность посещения маркеров. За невыполнение условия тест прерывается, и участник **получает 0 баллов**.
5. Сразу как робот заедет на маркер, функция `solve` должна записать в отправить (**смотри инструкцию**) строку «`P{имя маркера} OK`» (`P{имя маркера}` — вместо этой строки напишите имя маркера из задания, без кавычек), только в таком случае участник получит баллы. Если у маркера отсутствует обозначение ориентации, значит допускается любая ориентация робота на данном маркере.
   Пример строки с ответом.

   ```
   P3_SW OK
   P2_S OK
   P4_W OK
   P5_NE OK
   P16_W OK
   P8 OK
   P13_N OK
   P15 OK
   P16 OK
   ```

6. Считается, что робот заехал на марке тогда и только тогда, когда его центр находится пределах маркера (смотри пример ниже).



7. После завершения движения по маркерам робот должен заехать в зону `EXIT`. Сразу, как центр робота достигнет зоны `EXIT` задание считается выполненным.
8. Тест прерывается и участник **получает 0 баллов** при наезде на **черные стены**. Робот не должен пересекать стены. Робот может двигаться только по белым частям помещения и серым маркерам.

9. Тест прерывается и участник **получает 0 баллов**, если у участника не осталось времени на выполнение теста.

Участник должен написать решение в функции `solve`. В данном задании в качестве аргумента функции поступает объект симулятора, который позволяет управлять роботом и дает всю необходимую информацию о роботе (параметры кинематической модели, углы и скорости колес) и о карте (рендр помещения с роботами, изображение помещения с обозначениями).

Инструкцию по скачиванию, установке и запуску симулятора для начала выполнения задания можно найти по ссылке: https://gitlab.com/ovcharov.alex.o/nto_robotics. Здесь же лежат папки задачами, где подготовлены файлы `task*.py` и `task*.cpp`. Там также можно найти описание функции `solve` и минимально рабочий пример работы с системой для быстрого старта группы участников.

Решение

Решение может быть разным, допускается любое, удовлетворяющее правилам. Наиболее простым является:

1. Обрабатывая изображение комнаты, определяем координаты центра маркеров и сопоставляем с из названиями (из картинки комнаты с обозначениями).
2. Обрабатывая изображение комнаты, определяем координаты центра робота.
3. Обрабатывая изображение комнаты с обозначениями, определяем координаты из зоны `EXIT`.
4. С помощью алгоритма PRM строим связный граф из полученных точек.
5. Строим ПИД регулятор скоростью робота.
6. В цикле считаем путь по графу от текущего положения робота до целевой точки из задания.
7. В цикле движемся до желаемой точки с помощью ПИД регулятора. Для этого:
   7.1. обрабатывая изображение комнаты, постоянно определяем координаты центра робота и ориентацию;
   7.2. вычисляем вектор желаемой скорости робота на основе центра робота и построенного пути (хороший вариант интерполировать между точками в графе);
   7.3. считаем скорость робота на основе его кинематической модели.
8. После достижения желаемой точки отправляем результат о достижении точки (например `P13_W OK`) и движемся к следующему маркеру из теста.
9. После последнего маркера из теста строим путь по графу и движемся в точку из зоны `EXIT`.

Ссылка на тесты с симулятором и инструкцией по скачиванию, установке и запуску: https://gitlab.com/ovcharov.alex.o/nto_robotics.

Задача IV.3. Движение двух роботов по маркерам в свободном пространстве (25 баллов)

Темы: мобильная робототехника, дифференциальная платформа, одометрия, методы планирования траектории, компьютерное зрение, многоагентная система.

Условие

Данная задача отличается от предыдущей тем, что в нее добавлен еще один робот.

Введем основные обозначения, которыми будем пользоваться далее в условиях задач.

Участнику предоставляется симулятор, где в программе публикуется следующая информация:

1. Угол левого и правого колеса α_l, α_r каждой мобильной платформы.
2. Угловая скорость левого и правого колеса $\dot{\alpha}_l$, $\dot{\alpha}_r$ каждой мобильной платформы.
3. Изображение комнаты с мобильной платформой.
4. Изображение комнаты с основными обозначениями (названиями маркеров и зон).

Участник может управлять мобильной платформой, задавая напряжение на левом и правом двигателе.

Про изображение комнаты. Мы постарались смоделировать план помещения и представить, что на него постоянно сверху смотрит 2D камера. На изображении присутствуют:

- стены (черные);
- маркеры (серые квадраты, используются в заданиях);
- две мобильные платформы (робот 1 и робот 2);
- зона старта (`START`);
- зона окончания задания (`EXIT`).

Маркеры (`P1`, `P2`, `P3`, `P4` и т. д.) означают возможную зону (квадрат), в которую нужно будет приехать. Участнику предоставляется изображение помещения со всеми обозначениями и пример того, что участник увидит в симуляторе. Размер, цвет и форма маркеров во всех тестах одинаковые. На изображении комнаты с основными обозначениями присутствуют имена маркеров (`P1`, `P2` ...), они представлены только для примера. Участнику предстоит **самостоятельно присвоить имя маркеру**, на изображении представлен лишь пример того, как это может выглядеть.

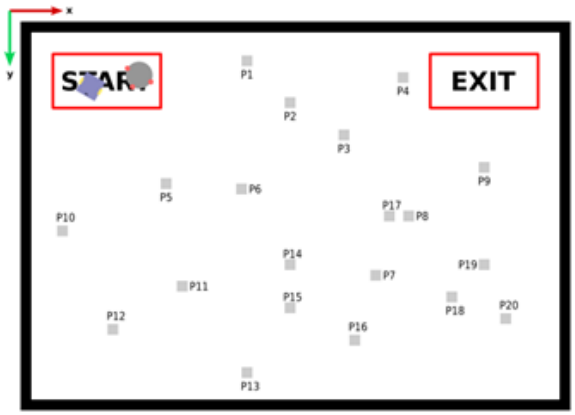


Рис. 9. Карта помещения с основными обозначениями

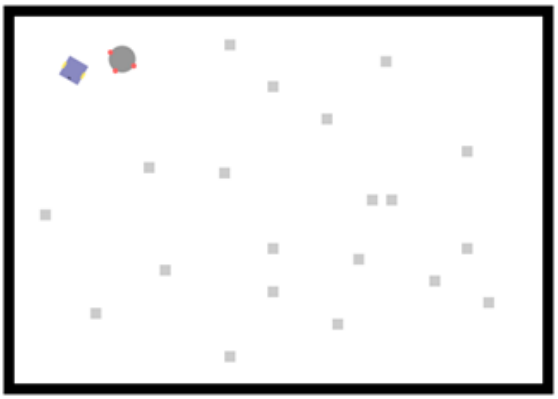
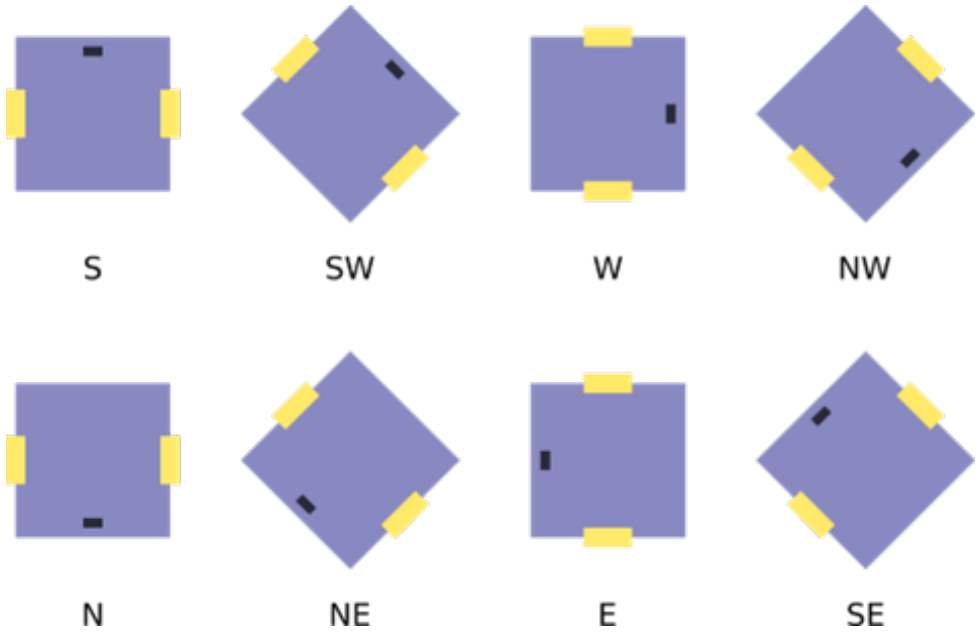


Рис. 10. Данные, которые видит участник в симуляторе

Изображение помещения с мобильной платформой внутри

В симуляторе карта обновляется с заданной частотой (**смотри параметры к задаче на следующем шаге**). Оси карты направлены также, как и в библиотеке OpenCV (x — вправо, y — вниз), точка отсчета находится в левом верхнем углу картинки. Также выделено восемь направлений: юг (S), север (N), запад (W), восток (E), юго-восток (SE), юго-запад (SW), северо-восток (NE), северо-запад (NW), они наглядно представлены на рисунке ниже.



Про мобильную платформу. Она представляет собой дифференциальную платформу с двумя колесами и располагается на стартовой позиции в помещении. Роботы представлены на рисунках 11 и 12, слева представлена графическая модель роботов, чей рендер отрисован на изображении помещения. Размер роботов во всех Тестах одинаковый, ширина примерно равна высоте.

Ниже, слева направо вводятся основные понятия и точки на роботе. Ориентация робота θ считается от горизонтальной оси x, положительное направление вращения робота совпадает с направлением движения стрелки часов (по часовой стрелке). Положение робота считается от левого верхнего угла изображения помещения.

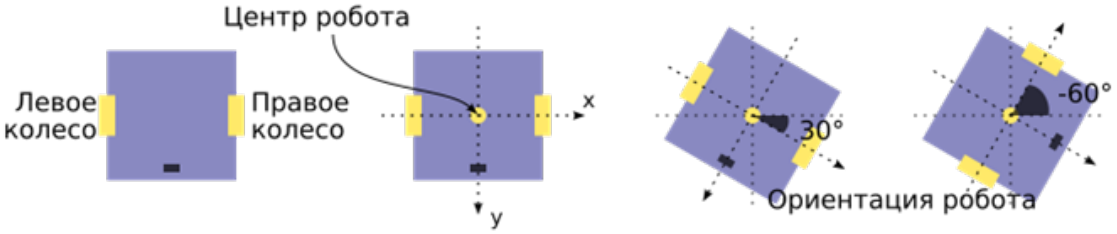


Рис. 11. Графическая модель первого робота и основные обозначения

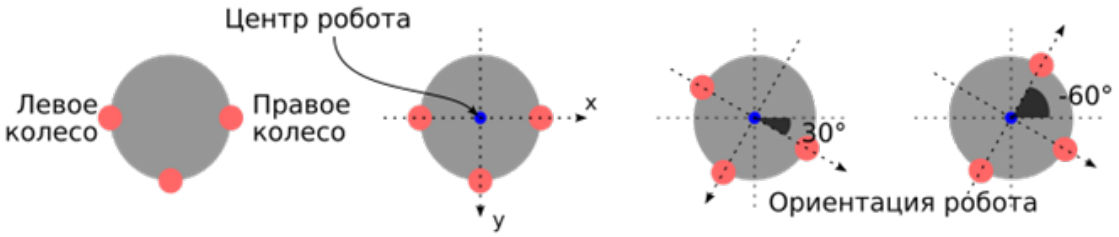


Рис. 12. Графическая модель второго робота и основные обозначения

Задание. После запуска симулятора робот расположен в зоне START (смотри рисунки 9–10) со случайной ориентацией. Задание разделено на несколько **карт**. В симуляторе вам предоставлена к решению только **первая карта**, есть также три судейских Карты с иным расположением маркеров (форма и размер не меняются).

Для каждой **карты** студент должен распознать и передать количество, имя маркеров и их координаты на изображении по шаблону «`P{№ маркера} {координата x} {координата y}`» (без кавычек). Пример:

```
5
P1 235 328
P2 15 92
P3 34 589
P4 858 322
P5 134 324
```

Важно сохранить далее эти имена, т. к. система будет использовать их для проверки. Для каждой **карты** генерируются **тесты**, в которых система создает строку со списком из N названий маркеров (имен, что участник передал ранее). Робот должен посетить каждый из этих маркеров. Участнику поступят следующие значения:

1. на первой строке указано количество меток, которые робот должен посетить,
2. номер метки и ориентация, с которой робот должен прийти в заданную метку.

Пример строки с заданием.

```
9
P3_SW P2_S P4_W P5_NE P16_W P8 P13_N P15 P16
P5 P8_W P2_E P1 P9_S P13_NE P16_N P10 P5
```

В случае успешного завершения Теста вы получаете полный балл. Предусмотрено начисление баллов за достижение маркеров (робот заехал на маркер с заданной ориентацией).

1. Сразу после старта симуляции запускается таймер. Время выполнения Теста ограничено.
2. После запуска, роботы должны рассчитать траекторию и начать движение к первому маркеру из списка.
3. После первого маркера из списка, роботы должны рассчитать траекторию и начать движение ко второму маркеру из списка и так далее.
4. Важно сохранять последовательность посещения маркеров. За невыполнение условия Тест прерывается и участник получает 0 баллов.

5. Сразу как роботы заедут на маркер, функция solve должна отправить строку строку «`P{robot1 marker number} P{robot2 marker number} OK`» (`P{robot1 marker number}` — номер маркера на котором сейчас первый робот, `P{robot2 marker number}` — аналогично номер маркера для второго робота). Роботы должны одновременно находится на целевых маркерах, только в таком случае участник получит баллы. Если у маркера отсутствует обозначение ориентации, значит допускается любая ориентация робота на данном маркере.
   Пример строки с ответом.

   ```
   P3_SW P1_N OK
   P2_S P3_S OK
   P4_W P2 OK
   P5_NE P4_W OK
   P16_W P16_W OK
   P8 P8 OK
   P13_N P13_N OK
   P15 P15 OK
   P16 P16_S OK
   ```

6. После завершения движения по маркерам робот должен заехать в зону `EXIT`. Сразу, как центр робота достигнет зоны `EXIT` задание считается выполненным.
7. Тест прерывается и участник **получает 0 баллов** при столкновении роботов друг с другом. Траектории роботов могут пересекаться, но **роботы не должны сталкиваться**.
8. Тест прерывается и участник **получает 0 баллов** при наезде на черные стены. Робот не должен пересекать стены. Робот может двигаться только по белым частям помещения и серым маркерам.
9. Тест прерывается и участник **получает 0 баллов**, если у участника не осталось времени на выполнение теста.

Участник должен написать решение в функции `solve`. В данном задании в качестве аргумента функции поступает объект симулятора, который позволяет управлять роботом и дает всю необходимую информацию о роботе (параметры кинематической модели, углы и скорости колес) и о карте (рендр помещения с роботами, изображение помещения с обозначениями).

Инструкцию по скачиванию, установке и запуску симулятора для начала выполнения задания можно найти по ссылке: https://gitlab.com/ovcharov.alex.o/nto_robotics. Здесь же лежат папки задачами, где подготовлены файлы `task*.py` и `task*.cpp`. Там также можно найти описание функции `solve` и минимально рабочий пример работы с системой для быстрого старта группы участников.

Решение

Решение может быть разным, допускается любое, удовлетворяющее правилам. Наиболее простым является:

1. Обрабатывая изображение комнаты, определяем координаты центра маркеров и сопоставляем с из названиями (из картинки комнаты с обозначениями).
2. Обрабатывая изображение комнаты, определяем координаты центра роботов.

3. Обрабатывая изображение комнаты с обозначениями, определяем координаты из зоны `EXIT`.
4. С помощью алгоритма PRM на белой зоне случайным образом генерируем много точек и строим связный граф из них и подготовленных координат (пункты 1, 2, 3).
5. Строим ПИД регулятор скоростью роботов.
6. В цикле считаем путь по графу от текущего положения робота до целевой точки из задания. В каждой точке проверяем, чтобы роботы не сталкивались.
7. В цикле движемся до желаемой точки с помощью ПИД регулятора. Для этого:
   7.1. обрабатывая изображение комнаты, постоянно определяем координаты центра робота и ориентацию;
   7.2. вычисляем вектор желаемой скорости робота на основе центра робота и построенного пути (хороший вариант интерполировать между точками в графе);
   7.3. считаем скорость робота на основе его кинематической модели;
8. После достижения желаемой точки отправляем результат о достижении точки (например `P13_W P14 OK`) и движемся к следующему маркеру из теста.
9. После последнего маркера из теста строим путь по графу и движемся в точку из зоны `EXIT`.

Ссылка на тесты с симулятором и инструкцией по скачиванию, установке и запуску: https://gitlab.com/ovcharov.alex.o/nto_robotics.

Задача IV.4. Движение по маркерам в ограниченном пространстве (30 баллов)

Темы: мобильная робототехника, дифференциальная платформа, одометрия, методы планирования траектории, обход препятствий, компьютерное зрение.

Условие

Введем основные обозначения, которыми будем пользоваться далее в условиях задач.

Участнику предоставляется симулятор, где в программе публикуется следующая информация:

1. Угол левого и правого колеса α_l, α_r мобильной платформы.
2. Угловая скорость левого и правого колеса $\dot{\alpha}_l$, $\dot{\alpha}_r$ мобильной платформы.
3. Изображение комнаты с мобильной платформой.
4. Изображение комнаты с основными обозначениями (названиями маркеров и зон).

Участник может управлять мобильной платформой, задавая напряжение на левом и правом двигателе.

Про изображение комнаты. Мы стараемся симулировать план помещения и представить, что на него постоянно сверху смотрит 2D камера. На изображении присутствуют:

- столы (желтые с черной окантовкой);

- стены (черные);
- маркеры (серые квадраты, используются в заданиях);
- мобильная платформа (робот);
- зона старта (`START`);
- зона окончания задания (`EXIT`).

Маркеры (`P1`, `P2`, `P3`, `P4` и т. д.) означают возможную зону (квадрат), в которую нужно будет приехать. Участнику предоставляется изображение помещения со всеми обозначениями и пример того, что участник увидит в симуляторе. Размер, цвет и форма маркеров во всех тестах одинаковые. На изображении комнаты с основными обозначениями присутствуют имена маркеров (`P1`, `P2` ...), они представлены только для примера. Участнику предстоит **самостоятельно присвоить имя маркеру**, на изображении представлен лишь пример того, как это может выглядеть.

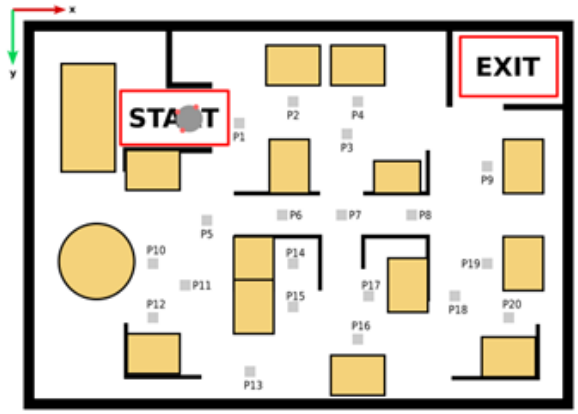


Рис. 13. Карта помещения с основными обозначениями

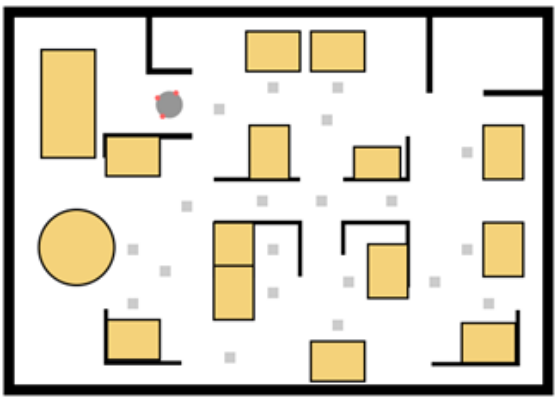
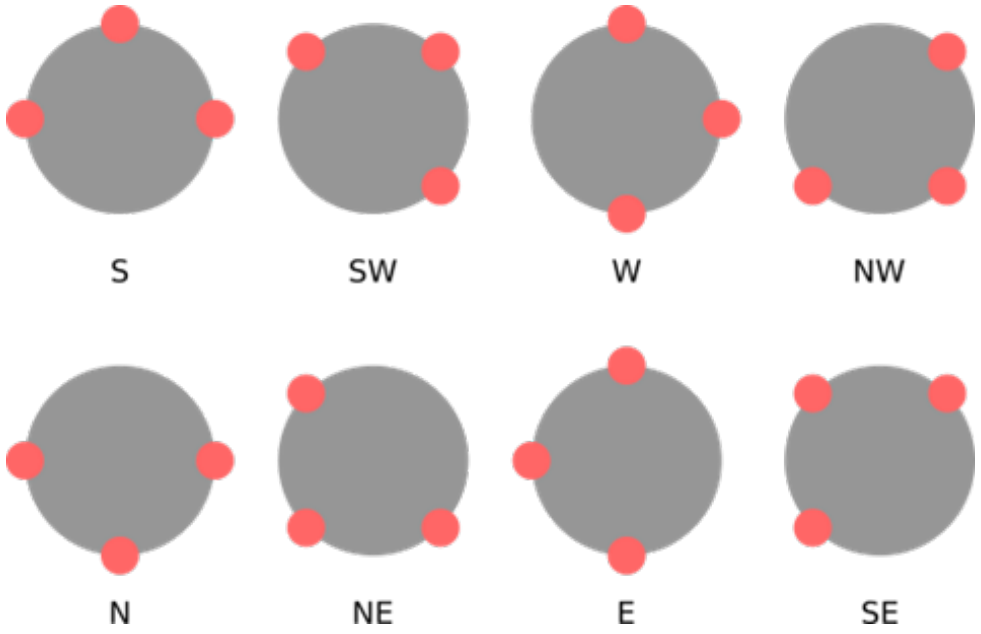


Рис. 14. Данные, которые видит участник в симуляторе

Изображение помещения с мобильной платформой внутри

В симуляторе карта обновляется с заданной частотой (**смотри параметры к задаче на следующем шаге**). Оси карты направлены также, как и в библиотеке OpenCV (x — вправо, y — вниз), точка отсчета находится в левом верхнем углу картинки. Также выделено восемь направлений: юг (S), север (N), запад (W), восток (E), юго-восток (SE), юго-запад (SW), северо-восток (NE), северо-запад (NW).



Про мобильную платформу. Она представляет собой дифференциальную платформу с двумя колесами и располагается на стартовой позиции в помещении. Робот имеет круглую форму, примерно как на рисунке 15, слева представлена графическая модель робота, чей рендер встречается на карте, далее справа вводятся основные понятия и точки на роботе. Ориентация робота считается от горизонтальной оси. Размер робота во всех Тестах одинаковый, ширина примерно равна высоте.

Ниже, слева направо вводятся основные понятия и точки на роботе. Ориентация робота θ считается от горизонтальной оси x, положительное направление вращения робота совпадает с направлением движения стрелки часов (по часовой стрелке). Положение робота считается от левого верхнего угла изображения помещения.

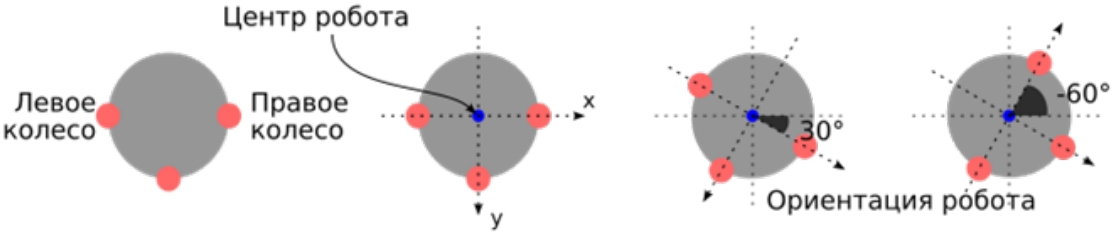


Рис. 15. Графическая модель робота и основные обозначения

Задание. После запуска симулятора робот расположен в зоне `START` (смотри рисунок 13–14) со случайной ориентацией. Задание разделено на несколько **карт**. В симуляторе вам предоставлена к решению только **первая карта**, есть также три судейских карты с иным расположением маркеров (форма и размер не меняются).

Для каждой **карты** студент должен распознать и передать количество, имя маркеров и их координаты на изображении по шаблону «`P{№ маркера} {координата x} {координата y}`» (без кавычек). Пример:

```
5
P1 235 328
P2 15 92
P3 34 589
```

```
P4 858 322
P5 134 324
```

Важно сохранить далее эти имена, т. к. система будет использовать их для проверки. Для каждой **карты** генерируются **тесты**, в которых система создает строку со списком из N названий маркеров (имен, что участник передал ранее). Робот должен посетить каждый из этих маркеров. Участнику поступят следующие значения:

1. на первой строке указано количество меток, которые робот должен посетить,
2. номер метки и ориентация, с которой робот должен прийти в заданную метку.

Пример строки с заданием.

```
9
P3_SW P2_S P4_W P5_NE P16_W P8 P13_N P15 P16
```

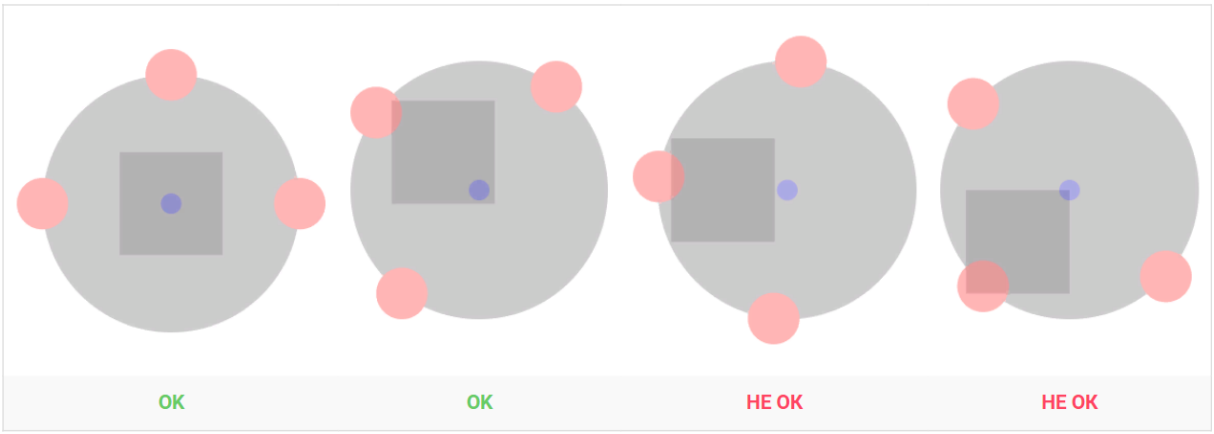
В случае успешного завершения теста вы получаете полный балл. Предусмотрено начисление баллов за достижение маркеров (робот заехал на маркер с заданной ориентацией).

1. Сразу после старта симуляции запускается таймер. Время выполнения теста ограничено.
2. После запуска робота он должен рассчитать траекторию и начать движение к первому маркеру.
3. После первого маркера из списка робот должен рассчитать траекторию и начать движение ко второму маркеру из списка и так далее.
4. Важно сохранять последовательность посещения маркеров. За невыполнение условия тест прерывается, и участник **получает 0 баллов**.
5. Сразу как робот заедет на маркер, функция `solve` должна записать в отправить (**смотри инструкцию**) строку «`P{имя маркера} OK`» (`P{имя маркера}` — вместо этой строки напишите имя маркера из задания, без кавычек), только в таком случае участник получит баллы. Если у маркера отсутствует обозначение ориентации, значит допускается любая ориентация робота на данном маркере.

   Пример строки с ответом.

   ```
   P3_SW OK
   P2_S OK
   P4_W OK
   P5_NE OK
   P16_W OK
   P8 OK
   P13_N OK
   P15 OK
   P16 OK
   ```

6. Считается, что робот заехал на марке тогда и только тогда, когда его центр находится пределах маркера (смотри пример ниже).



7. После завершения движения по маркерам робот должен заехать в зону `EXIT`. Сразу, как центр робота достигнет зоны `EXIT` задание считается выполненным.
8. Тест прерывается и участник **получает 0 баллов** при наезде на **черные стены**. Робот не должен пересекать стены. Робот может двигаться только по белым частям помещения и серым маркерам.
9. Тест прерывается и участник **получает 0 баллов**, если у участника не осталось времени на выполнение теста.

Участник должен написать решение в функции `solve`. В данном задании в качестве аргумента функции поступает объект симулятора, который позволяет управлять роботом и дает всю необходимую информацию о роботе (параметры кинематической модели, углы и скорости колес) и о карте (рендр помещения с роботами, изображение помещения с обозначениями).

Инструкцию по скачиванию, установке и запуску симулятора для начала выполнения задания можно найти по ссылке: https://gitlab.com/ovcharov.alex.o/nto_robotics. Здесь же лежат папки задачами, где подготовлены файлы `task*.py` и `task*.cpp`. Там также можно найти описание функции `solve` и минимально рабочий пример работы с системой для быстрого старта группы участников.

Решение

Решение может быть разным, допускается любое, удовлетворяющее правилам. Наиболее простым является:

1. Обрабатывая изображение комнаты, определяем координаты центра маркеров и сопоставляем с из названиями (из картинки комнаты с обозначениями).
2. Обрабатывая изображение комнаты, определяем координаты центра робота.
3. Обрабатывая изображение комнаты с обозначениями, определяем координаты из зоны `EXIT`.
4. С помощью алгоритма PRM строим связный граф из полученных точек.
5. Строим ПИД регулятор скоростью робота.
6. В цикле считаем путь по графу от текущего положения робота до целевой точки из задания.
7. В цикле движемся до желаемой точки с помощью ПИД регулятора. Для этого:
   7.1. обрабатывая изображение комнаты, постоянно определяем координаты центра робота и ориентацию;

7.2. вычисляем вектор желаемой скорости робота на основе центра робота и построенного пути (хороший вариант интерполировать между точками в графе);

7.3. считаем скорость робота на основе его кинематической модели.

8. После достижения желаемой точки отправляем результат о достижении точки (например `P13_W OK`) и движемся к следующему маркеру из теста.
9. После последнего маркера из теста строим путь по графу и движемся в точку из зоны `EXIT`.

Ссылка на тесты с симулятором и инструкцией по скачиванию, установке и запуску: https://gitlab.com/ovcharov.alex.o/nto_robotics.