Санкт–Петербургский государственный университет

ИВКИН Кирилл Андреевич

Выпускная квалификационная работа

Движение омниколесного робота по расчетной траектории

Уровень образования: бакалавриат
Направление 01.03.02 «Прикладная математика и информатика»
Основная образовательная программа СВ.5005.2016 «Прикладная математика, фундаментальная информатика и программирование»
Профиль «Прикладная математика, информатика и процессы управления»

Научный руководитель:
доцент, кафедра механики управляемого движения,
к.ф.- м.н., Шиманчук Дмитрий Викторович

Рецензент:
главный администратор CRIS-системы СПбГУ
к.ф.-м.н., Лепихин Тимур Андреевич

Санкт-Петербург
2020 г.

Содержание

Введение

Человек с давних времен пытался создать роботов. Еще в античности люди мечтали о гигантских бронзовых машинах, которые были бы способны помочь сражаться с врагами. Были даже попытки создания механических фигур, упоминания о которых были найдены в записках арабского изобретателя Аль-Джазари, датируемых примерно 1136-1206 годами. Первым, кто передставил чертеж человекоподобного робота, был Леонардо Да Винчи. И не смотря на столь давнее начало развития робототехники, только в 1921 году мехнизмы обрели четкий термин «робот». Исходя из истории можно заметить, что робот всегда был мечтой человека, которая сейчас воплощается в реальность.

В современном мире интеграция технологий в повседневную жизнь является одним из важнейших факторов развития человечества. В связи с этим в различных сферах деятельности человека все чаще применяется робототехника, а в особенности мобильные роботы, которые позволяют выполнять работу невозможную или опасную для человека, например в зонах химического заражения или в завалах шахт.

Автономным роботом будем называть робототехнический объект способный выполнять действия с высокой степенью автономии, основываясь на данных, получаемых им от датчиков или на шаблонах поведения.

Под автономным мобильным роботом будем понимать робототехнический объект, способный перемещаться самостоятельно в пространстве, основываясь на показаниях датчиков установленных на роботе или на получаемых извне данных, например по радиосвязи.

Постановка задачи

Будем решать задачу автономного движения четырехколесного омниколесного робота по расчетной траектории. Пусть имеется карта некоторого участка местности с отмеченными на ней препятствиями. Необходимо расчитать траекторию движения омниколесного робота из начальной точки в конечную в виде набора точек P_j, $j = \overline{0,n}$, обеспечивающего обход статических препятствий. Затем синтезировать закон управления мобильным роботом в виде $\omega_i = \omega_i(\rho_j, \alpha_j), i = \overline{1,4}$, где ω_i – угловая скорость i-ого колеса, α_j – курсовой угол на P_j, ρ_j – расстояние до P_j и реализовать алгоритм автономного движения, при котором робот переместится из начальной точки P_0 в конечную P_n.

Целью работы является построение программного комплекса по моделированию автономного движения омниколесного робота.

Обзор литературы

В ходе иследования был проведен обзор научной литературы, а также интернет-ресурсов.

Были изучены алгоритмы поиска пути, такие как: декомпозиция на ячейки, метод потенциального поля, метод вероятностной дорожной карты, а также метод RRT. Вышеупомянутые алгоритмы более подробно рассмотрены в источниках [1] и [2] списка литературы.

Было проведено исследование кинематической модели омниколесного робота с четырьмя независимо управляемыми колесами. Примеры такого робота: Kuka YouBot (Рис. 1), Segway RMP (Рис. 2), Uranus (Рис. 3). Эта модель активно использовалась в дальнейшей работе, она описана в источниках [3] и [4] списка литературы.

Был проведен анализ статьи [5], в которой представлен алгоритм управления мобильным омниколесным роботом по отклонению для движения по траектории.

Исследовано построение законов управления в форме обратной связи по курсовому углу и расстоянию до целевой точки для двухколесного мобильного робота. Детальное исследование вопроса проведено в статье [6]. Полученные результаты были в дальнейшем адаптированы для использования в изучаемой модели.

Для построения кинематической модели робота, а так же для реализации алгоритма автономного движения омниколесного робота по расчетной траектории были реализованы при помощи прикладных пакетов MATLAB & Simulink. Документация к программному обеспечению доступна на сайте [7].

Рис. 1: Kuka YouBot.

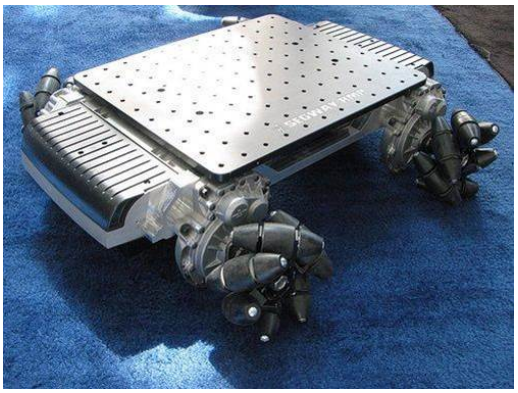

Рис. 2: Segway RMP.

Рис. 3: Uranus.

Глава 1. Алгоритмы поиска пути

Для решения поставленной задачи нам необходимо расчитать траекторию движения робота на основе карты некоторого участка местности с отмеченными препятствиями. Для этого в робототехнике применяются разнообразные алгоритмы построения пути, некоторые из них будут рассмотрены ниже.

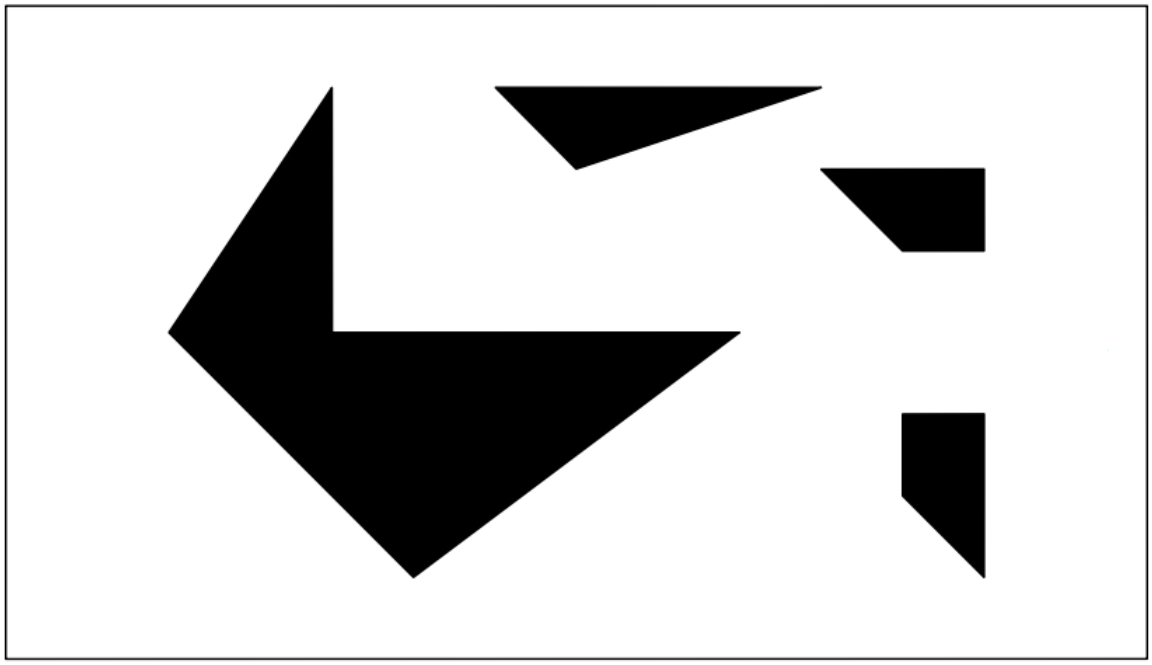

Рис. 4: Карта препятствий.

Под картой препятсвий (Рис. 4) понимается размеченная область, где черные области – это области препятствий, которых робот должен избегать, белая область – область поиска пути. В качестве карты также можно использовать матрицу размерности $m \times n$, где белой области ставится в соответствие нулевые элесенты матрицы, а черной – единичные.

1.1 Метод декомпозиции на ячейки

Основная идея метода декомпозиции на ячейки (Рис. 5) в разбиении свободного простанства на более мелкие треугольные и трапецеидальные секции, называемые ячейчками. После разделения пространства на ячейки строится граф связности, включающий в себя все возможные пути перемещения робота от начальной точки в конечную. Далее выбирается один из путей, оптимальный по какому либо критерию, указанному пользователем, например может выбираться самый короткий маршрут. Полученная последовательность вершин графа является искомой последовательностью точек $P_0 \dots P_n$.

Алгоритм 1 Метод декомпозиции на ячейки

1: Получаем конфигурационное пространство Q на основе карты местности.
2: Добавляем начальную и конечную точки в Q.
3: Производится декомпозиция Q на ячейки.
4: Определяются вершины графа связности на сторонах смежных ячеек.
5: Определяются вершины графа связности в геометрическом центре ячеек.
6: Формируются ребра графа связности.
7: Определяются последовательности ребер, соединяющих начальную и конечную вершины графа.
8: Выбирается оптимальная последовательность ребер.

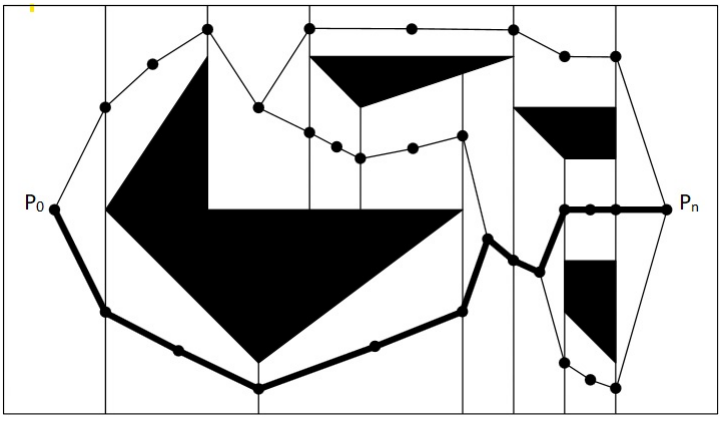

Рис. 5: Метод декомпозиции на ячейки.

1.2 Метод потенциального поля

В основе метода потенциальноо поля (Рис. 6) лежит идея представления робота в качестве частицы движущейся под воздействием некоторой функции потенциального поля. Конечная точка считается аттрактором, то есть областью притяжения, препятствия – репеллерами, то есть зонами отталкивания. После чего решается задача на поиск глобального минимума функции потенциального поля. Определенный в результате маршрут можно аппроксимировать последовательностью прямых, что позволит получить набор точек $P_0 \dots P_n$.

Алгоритм 2 Метод потенциального поля

1: Получаем конфигурационное пространство Q на основе карты местности.
2: Добавляем начальную и конечную точки в Q.
3: Задается аттрактивный компонент потенциального поля.
4: Задается репульсивный компонент потенциального поля.
5: Аттрактивный и репульсивный компоненты комбинируются.
6: Решается оптимизационная задача поиска глобального минимума функции потенциального поля.
7: Определяется маршрут, соединяющий начальную и конечную точки.

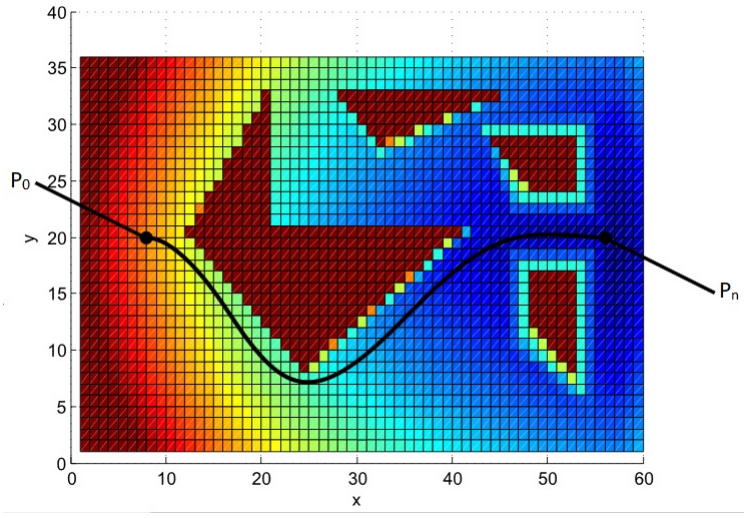

Рис. 6: Метод потенциального поля.

1.3 Метод вероятностной дорожной карты

Коллизией будем называть пересечение пути между двумя точками с препятствием. Идея метода вероятностной дорожной карты (Рис. 7) в использовании случайных выборок точек в свободном пространстве и создания графа связности на основе этих выборок. Затем по построенному графу строится маршрут из начальной точки в конечную. Полученная последовательность вершин является искомым набором точек $P_0 \dots P_n$. Данный метод достаточно прост в реализации.

Алгоритм 3 Метод вероятностной дорожной карты

1: Получаем конфигурационное пространство Q на основе карты местности.
2: Добавляем начальную и конечную точки в Q.
3: Производится случайная выборка n конфигураций.
4: Ближайшие узлы соединяются отрезками, если на пути между ними нет коллизий.
5: Пункты 2 и 3 повторяются до того момента пока не получится непрерывный путь из начальной точки в конечную или счетчик итераций не дойдет до конца.
6: Определяется маршрут, соединяющий начальную и конечную точки.

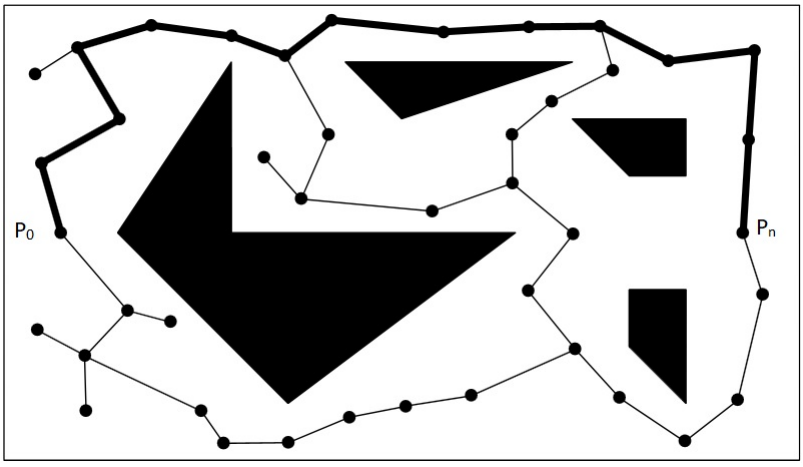

Рис. 7: Метод вероятностной дорожной карты.

1.4 Метод RRT

В методе RRT (Рис. 8) используется понятие случайного дерева. На первом шаге алгоритма в «пустое» дерево T добавляется вершина соответствующая начальной точке. На каждой итерации метода в дерево добавляется случайный узел, итерации происходят до тех пор пока не достигнута цель или не исчерпан счетчик итераций. Полученная последовательность узлов дерева является искомым набором точек $P_0 \dots P_n$. Качество построения маршрута, может варьроватся от случая к случаю.

Алгоритм 4 Шаг метода RRT

1: Случайным образом делаем выборку узла R.
2: Находим в T ближайший к R узел K.
3: По лучу от K до R делаем шаги на малую величину, пока не выполнится одно из условий:
   - наличие коллизии;
   - достигли максимального расстояния от K.
4: Если узел цели находится в пределах максимального расстояния и на пути от него к целевой точке нет коллизии, у нас есть решение.

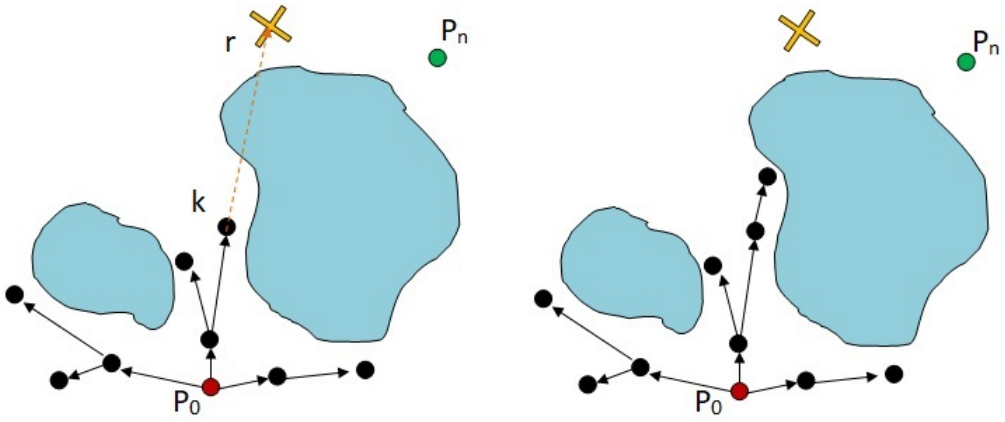

Рис. 8: Шаг метода RRT.

Глава 2. Решение задачи автономного движения

В данной главе будет рассмотрена математическая модель омниколесного робота, а также будут построены законы управления автономным мобильным роботом.

2.1 Математическая модель

Рассмотрим модель омниколесного робота с 4 независимо управляемыми колесами (Рис. 9). Особенность его конструкции заключается в специального вида колесах. Вместо протектора колеса прикрепляются свободно вращающиеся ролики под углом в 45 градусов. Такая конструкция позволяет роботу двигаться в любом направлении, варьируя угловые скорости поворота колес.

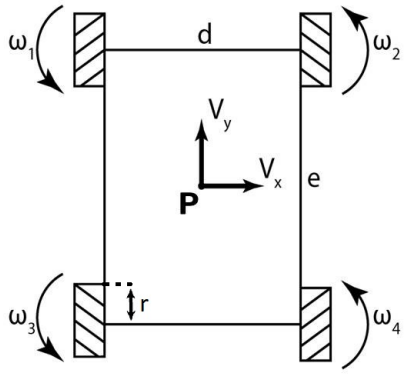

Рис. 9: Схема омниколесного робота.

Кинематическая модель робота описывается следующим матричным равенством [3]:

$$\begin{pmatrix} V_x \\ V_y \\ \omega \end{pmatrix} = r \begin{pmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{2(d+e)} & \frac{1}{2(d+e)} & -\frac{1}{2(d+e)} & \frac{1}{2(d+e)} \end{pmatrix} \begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{pmatrix},$$

где V_x – линейная скорость вдоль оси робота, V_y – линейная скорость пер-

пендикулярно оси робота, ω – угловая скорость робота, ω_i – угловая скорость i-ого колеса, $i = \overline{1,4}$, d – ширина робота, e – длина робота, r – радиус колес робота (см. Рис. 8).

Для того чтобы решать поставленную задачу нам необходимо решение обратной задачи кинематики для указанной модели. Оно представляется в виде [3]:

$$\begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{pmatrix} = \frac{1}{r} \begin{pmatrix} 1 & -1 & -\frac{d+e}{2} \\ 1 & 1 & \frac{d+e}{2} \\ 1 & 1 & -\frac{d+e}{2} \\ 1 & -1 & \frac{d+e}{2} \end{pmatrix} \begin{pmatrix} V_x \\ V_y \\ \omega \end{pmatrix}. \tag{1}$$

2.2 Синтез законов управления

Для навигации общей модели подвижного объекта в полярных координатах при переходе от точки P_{j-1} к точке P_j можно определить задачу в виде $\rho_j \to 0, \alpha_j \to 0$, где ρ_j – расстояние от полюса робота до целевой точки P_j, α_j – курсовой угол на целевую точку P_j. Закон управления будем строить в виде $\omega_i(\rho_j, \alpha_j)$, $i = \overline{1,4}$.

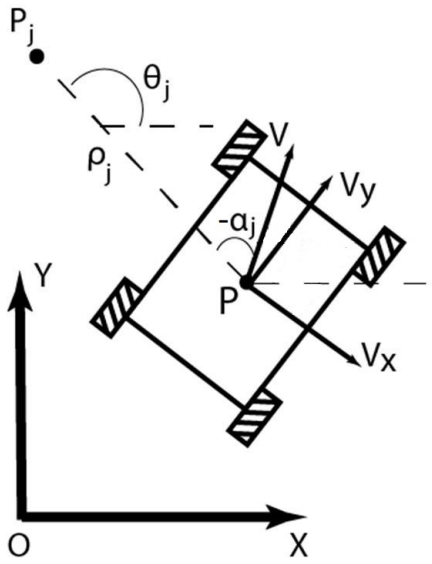

Рис. 10: Схема омниколесного робота в полярных координатах.

В полярных координатах навигация робота может быть определена следующей системой уравнений:

$$\begin{cases} \dot{\rho_j} = -V\cos(\alpha_j), \\ \dot{\alpha_j} = \omega + V\frac{\sin(\alpha_j)}{\rho_j}, \\ \dot{\theta_j} = -V\frac{\sin(\alpha_j)}{\rho_j}, \end{cases} \tag{2}$$

где V – величина линейной скорости полюса робота, θ_j – угол между осью O_X и вектором, направленным из центра робота к целевой точке, $j = \overline{1,n}$ (см. Рис. 10).

Из уравнений видно, что робот может управлятся с помощью значений угловой ω и линейной V скоростей. Будем искать такие значения V и ω, при которых выполняется условие поставленной задачи $\rho_j \to 0, \alpha_j \to 0$. Для этого воспользуемся математическим аппаратом функции Ляпунова. Согласно [6] в качестве функции Ляпунова рссмотрим квадратичную функцию, аргументами которой будут являться расстояние до цели и курсовой угол, то есть

$$L(\rho_j, \alpha_j) = \frac{1}{2}\rho_j^2 + \frac{1}{2}\alpha_j^2.$$

Из условия:

$$\dot{L}(\rho_j, \alpha_j)|_{(2)} = -V\rho_j\cos(\alpha_j) + \alpha_j(-\omega + V\frac{\sin(\alpha_j)}{\rho_j}) < 0$$

можно выбрать в качестве управляющего воздействия следующие значения скоростей:

$$\begin{cases} V = V_{ref} f(\rho_j)\cos(\alpha_j), \\ \omega = -k_\omega\alpha_j - \frac{V_{ref} f(\rho_j)\sin(\alpha_j)\cos(\alpha_j)}{\rho_j}, \end{cases} \tag{3}$$

где $V_{ref} > 0$, $k_\omega > 0$, функция $f(\rho_j)$ может быть выбрана в виде $sign(\rho_j)$, $\tanh(\rho_j)$ или $\frac{2}{1+e^{-a\rho}} - 1$, где $a \geq 1$. Вариант решения задачи, когда $f(\rho_j) = tanh(\rho_j)$, был рассмотрен в [6] и назывался тангенциальным избеганием. Выражение (3) решают задачу перемещения общей модели подвижного объекта в целевую точку P_j.

Выполняя подстановку значений угловой и линейной скоростей из (3)

в (2) получаем

$$\begin{cases} \dot{\rho}_j = -f(\rho_j)\cos^2(\alpha_j)V_{ref}, \\ \dot{\alpha}_j = -k_\omega \alpha_j \\ \dot{\theta}_j = -\frac{V_{ref} f(\rho_j)\sin(\alpha_j)\cos(\alpha_j)}{\rho_j}. \end{cases} \tag{4}$$

Далее, преобразуем (3):

$$\begin{cases} V_X = V_{ref} f(\rho_j)\cos(\alpha_j)\cos(\psi_j), \\ V_Y = V_{ref} f(\rho_j)\cos(\alpha_j)\sin(\psi_j), \\ \omega = -k_\omega \alpha_j - \frac{V_{ref} f(\rho_j)\sin(\alpha_j)\cos(\alpha_j)}{\rho_j}, \end{cases} \tag{5}$$

где V_X, V_Y – проекции вектора скорости на оси абсолютной системы координат, V_{ref} – допустимая линейная скорость робота, k_ω – коэффициент отвечающий за скорость изменения угловой скорости робота, подбирается для конкретной модели, $f(\rho_j)$ – функция зависящая от расстояния до точки и в идеальном случае являющаяся функцией $\mathrm{sign}(\rho_j)$, $\psi_j = \theta_j + \alpha_j$ – угол между направлением линейной скорости полюса P и осью O_X.

Заметим, что интегрируя уравнение (4) с начальными данными:

$$\begin{cases} \rho_j(0) = \rho_j^0, \\ \alpha_j(0) = \alpha_j^0, \\ \theta_j(0) = \theta_j^0, \end{cases}$$

и выполняя подстановку полученных функций $\rho_j(t)$, $\alpha_j(t)$ и $\theta_j(t)$ в (5) можно получить значения программных законов управления.

Далее, учитывая механику омниколесного робота, задачуавтономного движения можно рассмотривать в двух случаях: вектор линейной скорости полюса фиксирован относительно омниколесного робота или его направление изменяется в зависимости от угловых скоростей омниколес.

2.3 Первый режим работы робота

Для первого случая вектор $\{\frac{V_x}{V}, \frac{V_y}{V}\}$ – const. В данном режиме все слагаемые в формуле (2) сохраняют свой смысл и ориентация омниколесного робота определяется направлением вектора скорости. Автономное движение омниколесного робота, учитывая (5), осуществляется управляющими воздействиями:

$$\begin{cases} V_x = V_X \cos(\psi_j - \beta^*) + V_Y \sin(\psi_j - \beta^*), \\ V_y = -V_X \sin(\psi_j - \beta^*) + V_Y \cos(\psi_j - \beta^*) \\ \omega = -k_\omega \alpha_j - \frac{V_{ref} f(\rho_j) \sin(\alpha_j) \cos(\alpha_j)}{\rho_j}, \end{cases} \tag{6}$$

где $\beta^* = \text{atan2}(V_y, V_x)$ – угол между вектором скорости полюса P и направлением P_x. Для данного режима работы характерно то, что угловая скорость робота ω и угловая скорость вектора скорости робота V совпадают. Ниже представлена схема омниколесного робота для первого режима работы робота (Рис. 11).

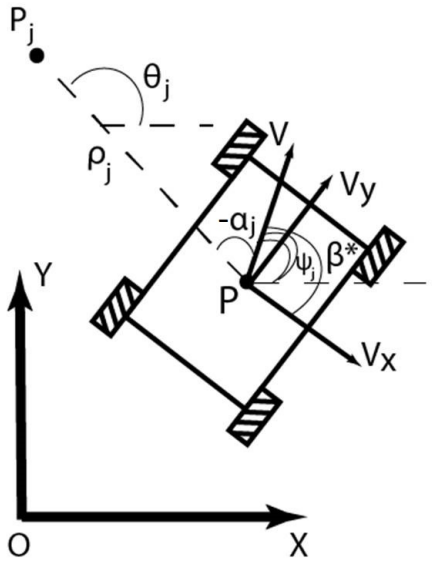

Рис. 11: Схема робота для первого режима работы.

2.4 Второй режим работы робота

Для второго режима работы робота (Рис. 12) необходимо ввести в рассмотрение параметр β, с помощью которого определяется ориентация системы координат P_{xy} (омниколесного робота) относительно абсолютного пространства O_{XY}. В этом случае система (4) переписывается в следующем виде:

$$\begin{cases} \dot{\rho} = -V_{ref} f(\rho_j), \\ \alpha_j \equiv 0, \\ \theta_j = c_j - const, \\ \omega = \dot{\beta}, \end{cases} \tag{7}$$

а автономное движение омниколесного робота, учитывая (5) и (7), осуществляется управляющими воздействиями:

$$\begin{cases} V_x = V_X \cos(\beta) + V_Y \sin(\beta), \\ V_y = -V_X \sin(\beta) + V_Y \cos(\beta), \\ \omega = \dot{\beta}. \end{cases} \tag{8}$$

В этом случае ветор скорости робота всегда направлен в целевую точку, а сама угловая скорость сожет быть определена некоторой заданной функцией, которой отвечает введенный нами параметр β. Нужно заметить, что в данном режиме работы угловая скорость вектора скорости робота и угловая скорость робота различны. Такое возможно реализовать, учитывая особенности омниколесного робота.

Подставляя полученные значения V_x, V_y и ω для одного из режимов работы робота (6) или (8) в решение обратной задачи кинематики (1), получим значения угловых скоростей колес робота $\omega_i = \omega_i(\rho_j, \alpha_j), i = \overline{1,4}$.

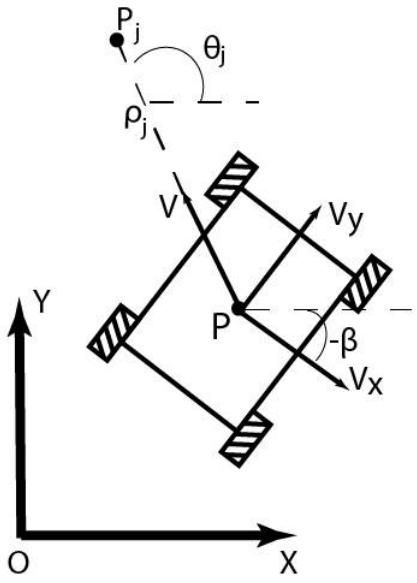

Рис. 12: Схема робота для второго режима работы.

Глава 3. Численный эксперимент

Для численного эксперимента, средствами MATLAB & Simulink, были были реализованы построение маршрута робота, а также модели для первого и второго режима работы робота.

3.1 Построение пути следования

Для построения пути следования мобильного робота будем использовать метод вероятностной дорожной карты, входящем в состав Robotics System Toolbox. Он является модификацией метода, рассмотренного в пункте 1.3 данной работы. Отличие от предыдущего метода заключается в том, что в используемом для эксперимента выборка из n точек генерируется один раз, а затем строятся все возможные пути, попарно соединяющие точки на заданном расстоянии Δ друг от друга, если на пути между ними нет коллизий. Таким образом у модифицированного алгоритма одна итерация.

Для построенния траектории была взята квадратная карта длина и ширина которой 0.5 метра с отмеченными на ней препятствиями.

Начальная точка: (0.5, 0.5). Конечная точка: (0.5, 3.5). Размер выборки: 200 точек. $\Delta = \infty$.
Был построена следующая траектория (Рис. 13):

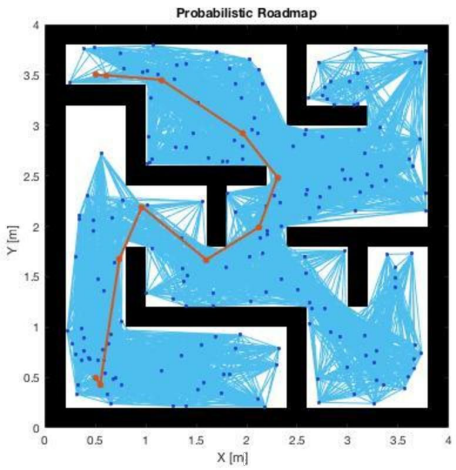

Рис. 13: Расчетная траектория движения робота.

Далее несколько модифицируем полученный маршрут, а именно исключаем из полученного набора точек вторую и предпоследнюю.

Таким образом, был получен набор точек P_j, $j = \overline{0, n}$, которые будем использовать для последующих вычислений.

3.2 Первый режим работы робота

Рассмотрим результаты компьютерного моделирования первого режима работы робота. В модели используется траектория полученная в пункте 3.1 данной работы. Целевая точка считается достигнутой если $\alpha_j \leq \epsilon$, $\rho_j \leq \delta$.

Параметры модели для численного эксперимента:
$e = 0.2$м, $d = 0.1$м, $r = 0.03$м, $k_\omega = 0.1$, $V_{ref} = 0.01$м/с, $\epsilon = 10^{-6}$м, $\delta = 10^{-4}$м, $\beta^* = 0.3805$, $f(\rho_j) = tanh(\rho_j)$.

Рассмотрим полученные результаты:

1. График маршрута омниколесного робота (Рис. 14), здесь синей линией отмечена расчетная траектория, а красной траектория движения робота. Красные точки на графике – точки перехода от движения к P_j к движению к P_{j+1}.

2. Графики изменения расстояния до целевой точки и курсового угла на нее (Рис. 15).

3. Графики изменения проекций линейной скорости робота на абсолютные оси координат и угловой скорости робота (Рис. 16).

4. Графики изменения угловых скоростей колес робота (Рис. 17).

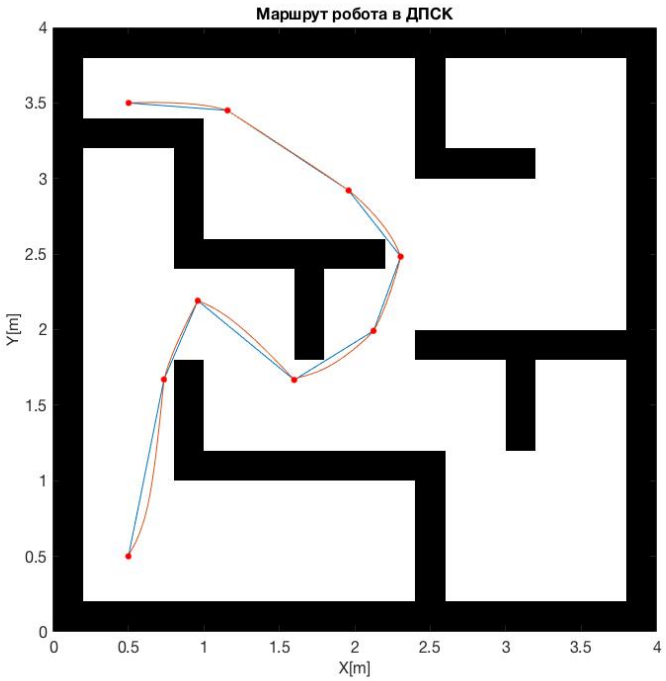

Рис. 14: Маршрут робота в ДПСК.

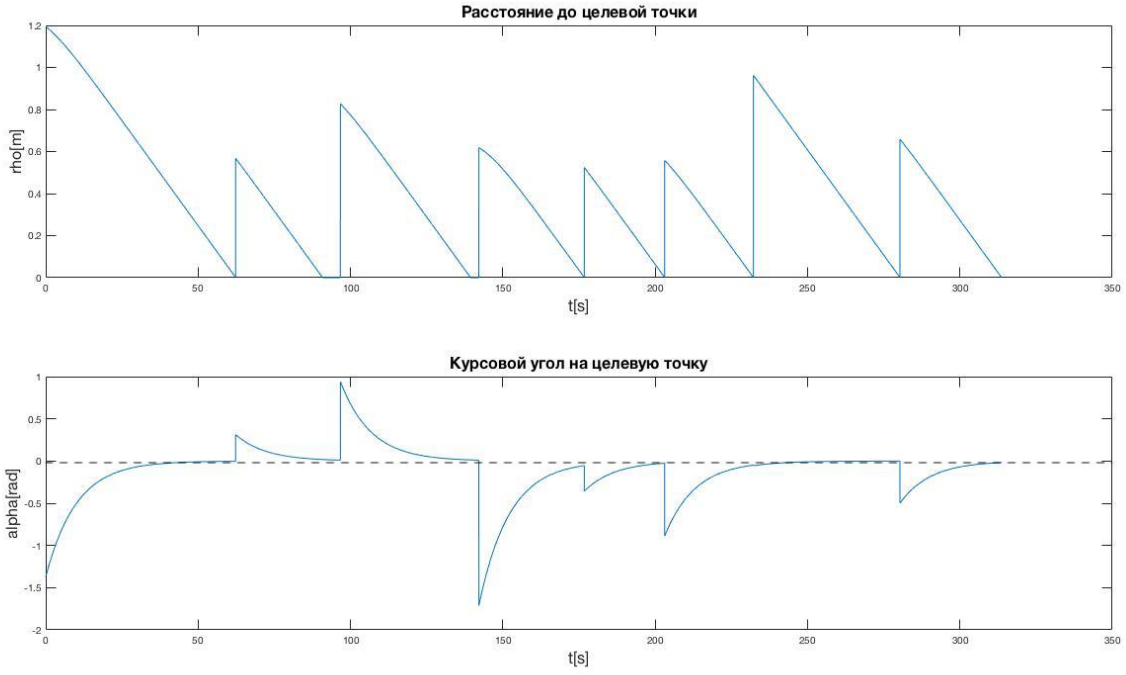

Рис. 15: Графики для ρ и α.

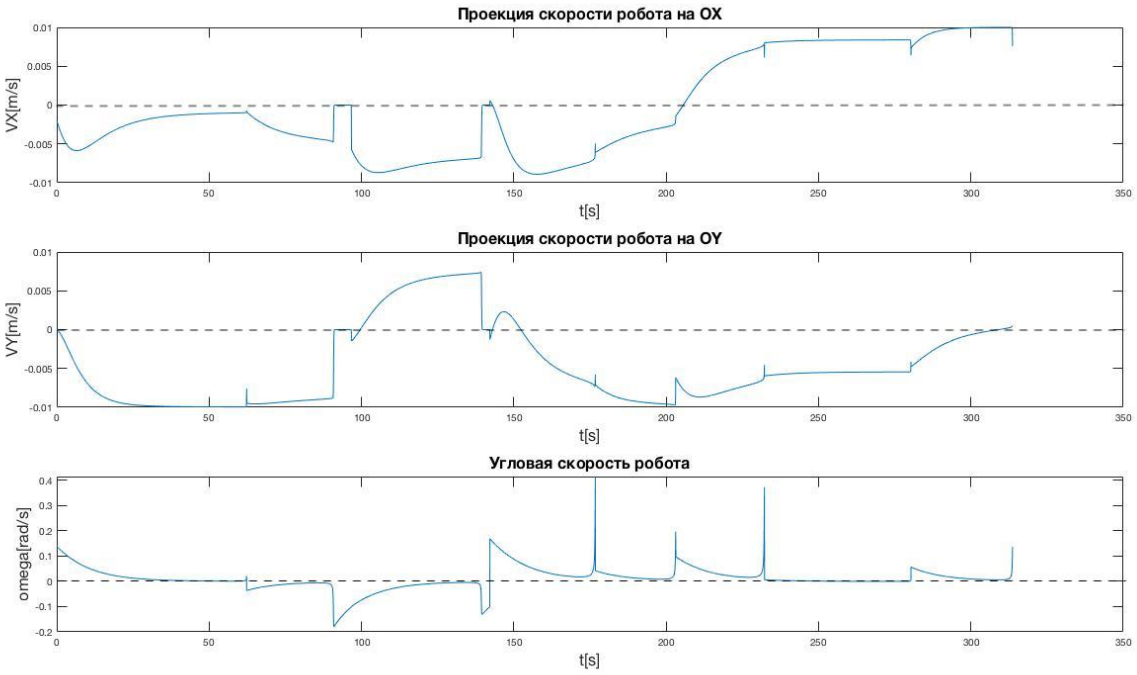

Рис. 16: Графики для V_X, V_Y и ω.

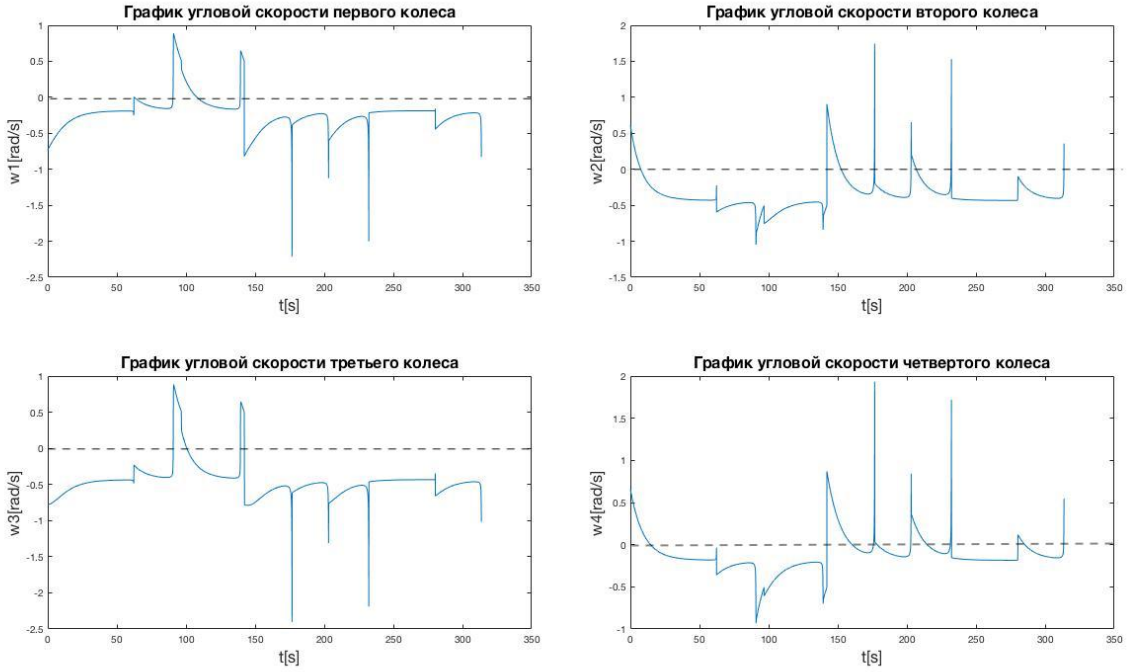

Рис. 17: Графики угловых скоростей колес робота.

3.3 Второй режим работы робота

Перейдем к моделированию второго режима работы робота. В модели также используется траектория постоенная в пункте 3.1. Целевая точка считается достигнутой если $\rho_j \leq \delta$. Условие для альфа не требуется, это следует из формулы (7).
Параметры модели для численного эксперимента:

$e = 0.2$м, $d = 0.1$м, $r = 0.03$м, $k_\omega = 0.1$, $V_{ref} = 0.01$м/с, $\delta = 10^{-4}$м, $\beta = 5$, $f(\rho_j) = tanh(\rho_j)$.

Рассмотрим полученные результаты:

1. График маршрута омниколесного робота (Рис. 18), здесь синей линией отмечена расчетная траектория, а красной траектория движения робота. Красные точки на графике – точки перехода от движения к P_j к движению к P_{j+1}.

2. Графики изменения расстояния до целевой точки (Рис. 19).

3. Графики изменения проекций линейной скорости робота на абсолютные оси координат и угловой скорости робота (Рис. 20).

4. Графики изменения угловых скоростей колес робота (Рис. 21).

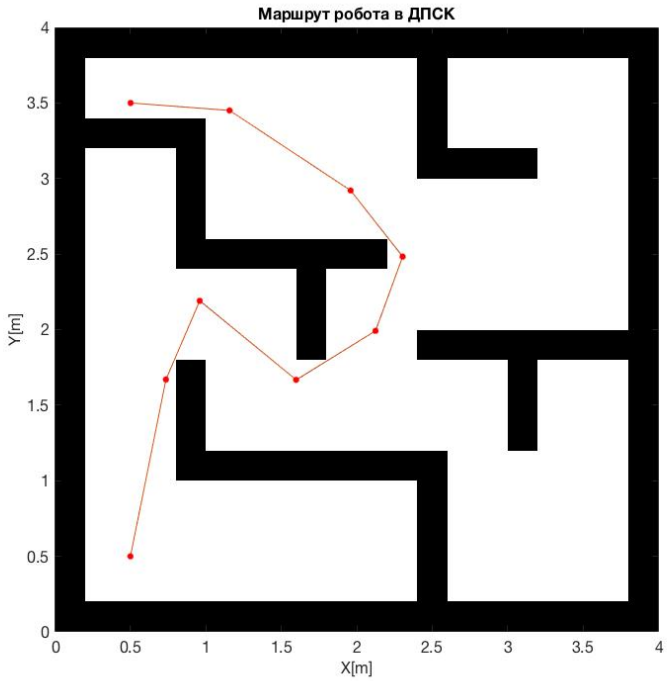

Рис. 18: Маршрут робота в ДПСК.

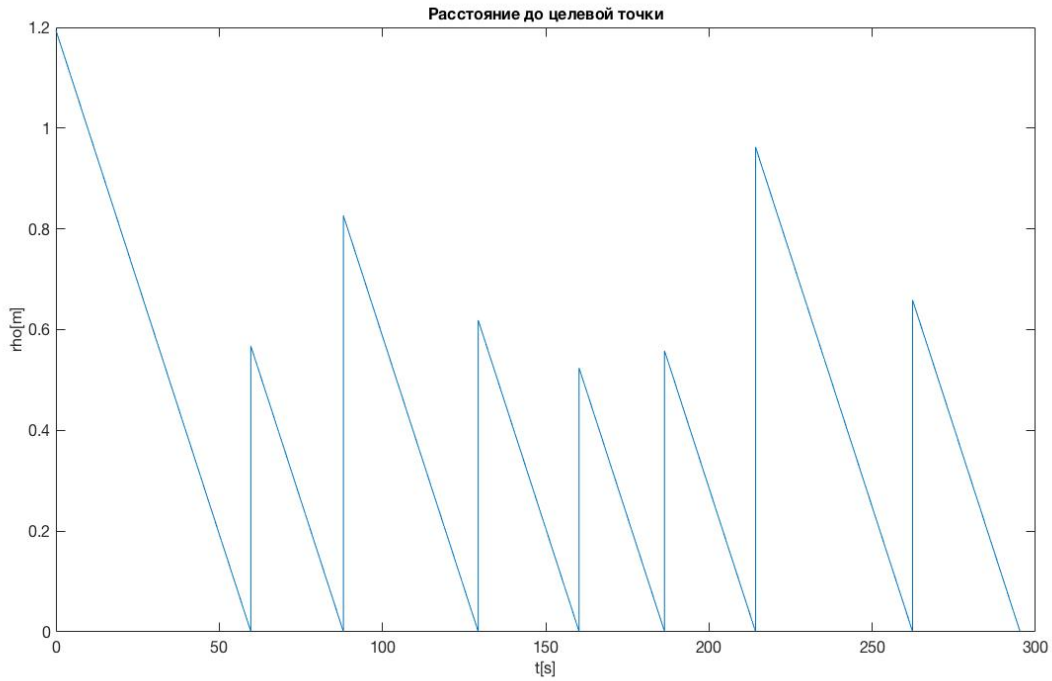

Рис. 19: График для ρ.

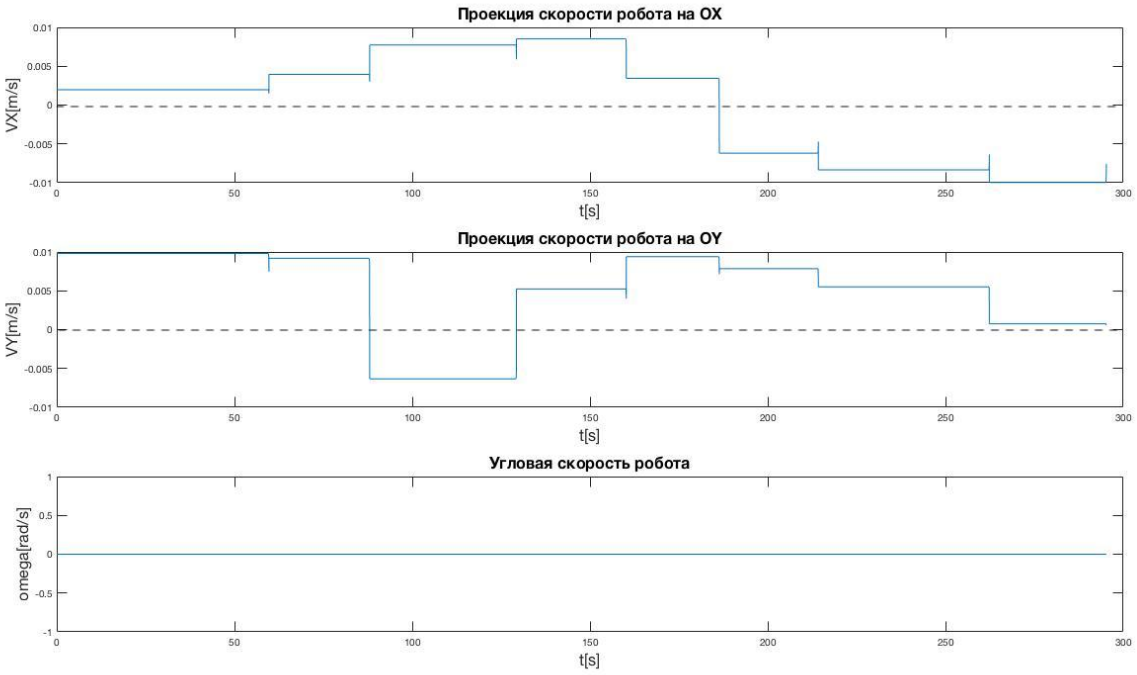

Рис. 20: Графики для V_X, V_Y и ω.

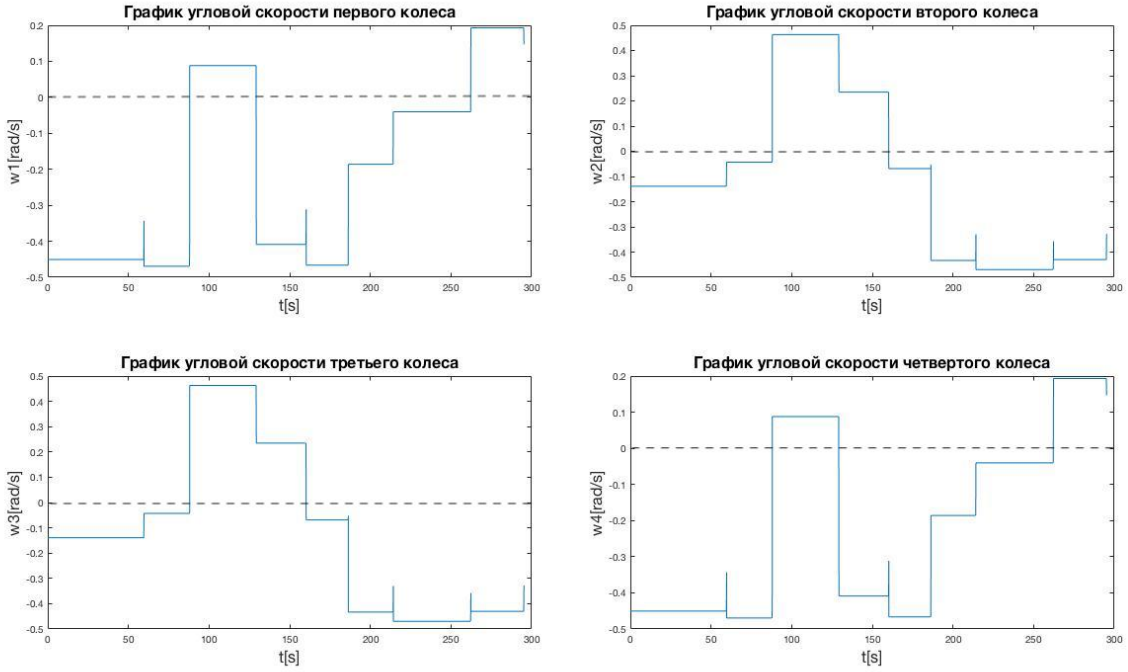

Рис. 21: Графики угловыч скоростей колес робота.

Заключение

В процессе работа были решены следующие задачи:

1. Рассмотрены алгоритмы поиска пути.

2. Разработаны два режима работы омниколесного робота.

3. Построены законы управления в форме обратной связи $\omega_i = \omega_i(\rho_j, \alpha_j), i = \overline{1,4}$ для двух режимов работы робота, которые обеспечивают автономное движение омниколесного мобильного робота вдоль расчетной траектории, согласно выделенному множеству её точек P_j, $j = \overline{0,n}$.

4. Модифицированным методом вероятностной дорожной карты расчитана траектория движения омниколесного робота в виде набора точек P_j, $j = \overline{0,n}$.

5. Средствами MATLAB & Simulink реализован программный комплекс по моделированию автономного движения омниколесного робота по расчетной траектории. В приложении к работе приведен программный код на языке Matlab построения маршрута движения (Приложение 1), моделирования первого (Приложение 2) и второго (Приложение 3) режимов работы омниколесного робота.

Стоит заметить, что для второго случая реализации движения параметр β определяет желаемую ориентацию, которая может быть обусловлена конкретной постановкой задачи, омниколесного робота в абсолютном пространстве. Идентификация параметров V_{ref} и k_ω для реальной модели робота является отдельной задачей и при проектировании может быть проведена с помощью численных экспериментов.

Список литературы

[1] LaValle S.M. «Planning Algorithms». Cambridge University Press. 2006. C. 228,229.

[2] Борисов О.И., Громов В.С., Пыркин А.А. «Методы управления робототехническими приложениями. Учебное пособие». СПб.: Университет ИТМО, 2016. С. 41–46.

[3] Thomas Bräunl.«Embedded Robotics: Mobile Robot Design and Applications with Embedded Systems». Springer Science & Business Media. 2008. P. 147–156.

[4] J. Agulló, S. Cardona, J. Vivancos.«Kinematics of vehicles with directional sliding wheels». Mechanism and Machine Theory, Volume 22, Issue 4, 1987, P. 295–301.

[5] Ю. Л. Караваев, С. А. Трефилов, «Дискретный алгоритм управления по отклонению мобильным роботом с омниколесами», Нелинейная динам., 2013, том 9, номер 1, 91–100

[6] Ferreira A., Vassallo R.F., Pereira F.G., Filho T.F.B., Filho M.S. «An approach to avoid obstacles in mobile robot navigation: the tangential escape»// Controle y Automacao. 2008. V.19. N 4. P. 395–405.

[7] Сетевой ресурс: `https://www.mathworks.com/help/` – документация к MATLAB & Simulink. Дата обращения: 18.03.2020.

[8] Маркеев А.П. «Теоретическая механика». 2-е изд. Учебник для университетов. Москва: ЧеРо. 1999, 572 стр.

[9] Сетевой ресурс: `https://habrastorage.org/getpro/habr/post_images/16b/547/8d4/16b5478d425906c486d29153e2055e73.png` – Рисунок 8. Дата обращения: 20.04.2020

[10] Сетевой ресурс: `https://i.pinimg.com/564x/4f/a0/dc/4fa0dcee7ed5413ea8f3ca1e5051273c.jpg` – Рисунок 1. Дата обращения: 20.04.2020.

[11] Сетевой ресурс: http://www.membrana.ru/storage/img/n/n0w.jpg – Рисунок 2. Дата обращения: 20.04.2020.

[12] Сетевой ресурс: https://www.cs.cmu.edu/~gwp/robots/Uranus.jpg – Рисунок 3. Дата обращения: 20.04.2020.

Приложение 1

```
% creating obstacle map
testMap = [...
    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1;
    1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1;
    1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1;
    1 1 1 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1;
    1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 1;
    1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1;
    1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1;
    1 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1;
    1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1;
    1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1;
    1 0 0 0 0 0 0 0 1 0 0 0 1 1 1 1 1 1 1 1;
    1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1;
    1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1;
    1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1;
    1 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1;
    1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1;
    1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1;
    1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1;
    1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1;
    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1];

start = [0.5 0.5]; % start point
finish = [0.5 3.5]; % end point
map = binaryOccupancyMap(testMap,5); % creates binary occupancy
% map 4x4 meters

% pathfinding
rngState = rng;
prm = mobileRobotPRM(map,200);
path = findpath(prm,start,finish);
rng(rngState);
path = findpath(prm,start,finish);
show(prm)
% path points in mm
Xprm = path(:,1)*1000;
Yprm = path(:,2)*1000;
% deleteting the 2nd and (n-1)th points for better path
Xprm(2) = [];
Yprm(2) = [];
Xprm(end-1) = [];
Yprm(end-1) = [];
```

Приложение 2

```
eps = 10e-3; % exit condition for alpha
delta = 10e-1; % exit condition for rho
e = 200; % robot lenght in mm
d = 100; % robot width in mm
r = 30; % radius of robot wheels in mm

Vref = 10; % max velocity in mm/s

komega = 0.1;

% velocity vector direction
Vx = 5; % mm/s
Vy = 2; % mm/s

beta_star = atan2(Vy, Vx); %rad

% matrix for conversion of linear velocity of robot and its angle
% velocity to angle velocity of wheels

invMotionMatrix = (1/r)*[1 -1 (e + d)/-2;...
                         1 1 (e + d)/2;...
                         1 1 (e + d)/-2;...
                         1 -1 (e + d)/2];

% initiation of motion
n = max(size(path))-3;
w1 = [];
w2 = [];
w3 = [];
w4 = [];
rho = [];
alpha = [];
VX = [];
VY = [];
omega = [];
theta = [];
rhoT = 0;
alphaT = 0;
thetaT = 0;

% motion from P_j to P_j+1
for i = 1:n
    rho0 =  sqrt((Xprm(i+1)-(Xprm(i)-rhoT*cos(thetaT)))^2 ...
                +(Yprm(i+1)-(Yprm(i)-rhoT*sin(thetaT)))^2);
```

```
    theta0 = atan2((Yprm(i+1)-(Yprm(i)-rhoT*sin(thetaT))),...
                   (Xprm(i+1)-(Xprm(i)-rhoT*cos(thetaT))));
    alpha0 = thetaT + alphaT - theta0;

    out = sim('eqns1st',inf);
    w1 = [w1; out.w1];
    w2 = [w2; out.w2];
    w3 = [w3; out.w3];
    w4 = [w4; out.w4];
    rho = [rho; out.rho];
    alpha = [alpha; out.alpha];
    VX = [VX;out.VX];
    VY = [VY;out.VY];
    omega = [omega;out.omega];
    theta = [theta;out.theta];
    rhoT = rho(end);
    alphaT = alpha(end);
    thetaT = theta(end);
end

n = size(w1);
n = n(1);
t = 0:0.001:n/1000 - 0.001; % time vector

% calculating moments of time when robot switch movement direction
switchPoints = 1;
for i=1:size(VX)-1
    temp = rho(i)-rho(i+1);
    if  temp <= 0
        switchPoints = [switchPoints;i];
    end
end
switchPoints = [switchPoints;n];

% calculating robot's path in OXY
X = Xprm(1);
Y = Yprm(1);
k=1;
for i=1:size(VX)-1
    X = [X;Xprm(k+1) - rho(i)*cos(theta(i))];
    Y = [Y;Yprm(k+1) - rho(i)*sin(theta(i))];
    if i == switchPoints(k+1)
        k = k+1;
    end
end
```

Приложение 3

```
delta = 10e-1; % exit condition for rho
e = 200; % robot lenght in mm
d = 100; % robot width in mm
r = 30; % radius of robot wheels in mm

Vref = 10; % max velocity in mm/s

komega = 0.1;

b = @(t) 5; % beta

% matrix for conversion of linear velocity of robot and its angle
% velocity to angle velocity of wheels

invMotionMatrix = (1/r)*[1 -1 (e + d)/-2;...
                         1 1 (e + d)/2;...
                         1 1 (e + d)/-2;...
                         1 -1 (e + d)/2];

% initiation of motion
n = max(size(path))-3;
w1 = [];
w2 = [];
w3 = [];
w4 = [];
rho = [];
alpha = [];
VX = [];
VY = [];
omega = [];
theta = [];
rhoT = 0;
alphaT = 0;
thetaT = 0;

% motion from P_j to P_j+1
for i = 1:n
    rho0 =  sqrt((Xprm(i+1)-(Xprm(i)-rhoT*cos(thetaT)))^2 ...
                +(Yprm(i+1)-(Yprm(i)-rhoT*sin(thetaT)))^2);
    theta0 = atan2((Yprm(i+1)-(Yprm(i)-rhoT*sin(thetaT))),...
                   (Xprm(i+1)-(Xprm(i)-rhoT*cos(thetaT))));
    alpha0 = 0;

    out = sim('eqns2nd',inf);
```

```
    w1 = [w1; out.w1];
    w2 = [w2; out.w2];
    w3 = [w3; out.w3];
    w4 = [w4; out.w4];
    rho = [rho; out.rho];
    VX = [VX;out.VX];
    VY = [VY;out.VY];
    omega = [omega;out.omega];
    theta = [theta;out.theta];
    rhoT = rho(end);
    thetaT = theta(end);
end

n = size(w1);
n = n(1);
t = 0:0.001:n/1000 - 0.001; % time vector

% calculating moments of time when robot switch movement direction
switchPoints = 1;
for i=1:size(VX)-1
    temp = rho(i)-rho(i+1);
    if  temp <= 0
        switchPoints = [switchPoints;i];
    end
end
switchPoints = [switchPoints;n];

% calculating robot's path in OXY
X = Xprm(1);
Y = Yprm(1);
k=1;
for i=1:size(VX)-1
    X = [X;Xprm(k+1) - rho(i)*cos(theta(i))];
    Y = [Y;Yprm(k+1) - rho(i)*sin(theta(i))];
    if i == switchPoints(k+1)
        k = k+1;
    end
end
```