Научная статья
Original article
УДК 004.031

Система Управления Коллаборативным Роботом На Основе ROS

ROS-BASED COLLABORATIVE ROBOT CONTROL SYSTEM

Сун Бисянь, студент факультета специального машиностроения, Московский государственный технический университет имени Н. Э. Баумана, г. Москва. 13390773394@163.com

Ван Вэньюэ, студент факультета специального машиностроения, Московский государственный технический университет имени Н. Э. Баумана, г. Москва. 18761875608@163.com

Song Bixian, student of the Faculty of Special Engineering, Bauman Moscow State Technical University, Moscow. 13390773394@163.com

Wang Wenyue, student of the Faculty of Special Engineering, Bauman Moscow State Technical University, Moscow. 18761875608@163.com

Аннотация. Для реализации управления коллаборативным роботом изучается система управления коллаборативным роботом. В целях обеспечения надежности и устойчивости системы в реальном времени, мы предлагаем построить систему управления коллаборативным роботом на ПК на базе системы Ubuntu с ROS и CAN коммуникацией. В заключение, практическое применение системы управления коллаборативным роботом проверяется с помощью имитационного эксперимента по управлению роботом. Результаты экспериментов показывают, что система управления коллаборативным роботом обладает основными способностями для планирования траектории движения коллаборативного робота и может

устанавливать связь между главным компьютером и коллаборативным роботом для управления роботом. Система управления также является модульной, очень портативной, с четкой структурой и низкой задержкой.

Abstract. In order to realize the control of the cooperative robot, the control system of the cooperative robot was studied. On the premise of ensuring the robustness and real-time of the system, it is proposed to build a control system based on Ubuntu system combined with Robot Operating System (ROS), and use Controller Area Network (CAN) communication to build a collaborative robot. Finally, through the simulation experiment and the physical robot control experiment, the application effect of the collaborative robot control system is verified. The experimental results show that the cooperative robot control system has the basic working ability to control the coordinated robot for path planning, and can well establish the communication between the upper and lower machines and control the cooperative robot. At the same time, the control system has the characteristics of modularity, high portability, clear frame, and low delay.

Ключевые слова: коллаборативный робот, система ROS, проектирование системы управления, алгоритм RRT, коммуникационный интерфейс

Keywords: collaborative robot, ROS system, control system design, algorithm RRT, communication interface

Введение

По мере роста популярности использования роботов и повышения требований к точности, увеличивается количество разработок аппаратуры и систем управления для роботов. Увеличение количества различных типов роботов привело к разработке различных систем управления.

В настоящее время, хотя существует множество компаний, занимающихся исследованиями и разработками в области робототехники, системы управления, которые они разрабатывают, все нацелены на отдельный робот и не являются универсальными или применимыми на рынке, что не способствует развитию систем управления робототехникой. Поэтому необходимость разработки системы управления, которая обладает универсальностью и подходит для большинства современных роботов, является насущной потребностью рынка.

При разработке систем управления для современных роботов можно в полной мере рассматривать несколько платформ разработки, таких как Microsoft visual, Matlab, MRDS и т.д. Microsoft visual [1] подходит для разработки в среде Windows и является более комплексным, но процесс разработки сложен и часто чреват ошибками; Matlab [2] в основном используется для экспериментов по моделированию, но в нем трудно реализовать управление физическим роботом. И поскольку эти программные обеспечения инкапсулируют некоторые алгоритмы, например, OpenCV, Ompl и др. Поэтому необходимо найти платформу разработки, которая совместима с различными алгоритмами и проста в использовании для нескольких платформ.

Для этого в данной работе была создана платформа системы управления на основе операционной системы Robot Operating System (ROS) [3], а также проведены соответствующие эксперименты по моделированию и физическому управлению.

1. Настройка среды конфигурирования ROS

Robot Operating System (ROS) - это система разработки систем управления роботами. Он включает в себя 3D-реконструкцию, динамическое моделирование, моделирование среды в реальном времени, физическое управление и другие функции, позволяющие максимально повысить требования к разработке роботов [4].

ROS – это внутренняя операционная система, встроенная в Ubuntu, но не имеющая отношения к управлению процессами системы. Он предоставляет различные функциональные пакеты, а затем устанавливает коммуникацию между пакетами через одноранговый режим для создания платформы управления для моделирования и физического управления роботом, что порождает такие основные понятия, как "нода", "топик", "сервис" и "сообщение". Эти основные понятия, как показано на рисунке 1, используются для предоставления данных системе управления посредством взаимодействия между ними.

1.1 Нода (Node)

Ноды – это различные исполняемые файлы в ROS, которые взаимодействуют друг с другом через топики и сервисы. Они используются для повышения

эффективности и ремонтопригодности системы. Ноды являются уникальными в системе, и имена нод не могут повторяться. Это позволяет связи между нодами становиться уникальной.

1.2 Топик (Topic)

Топик – это магистральная шина для передачи данных между нодами. Издатель, отправляющий и подписчик, получающий топик, не обязаны знать о существовании друг друга. Топик может иметь несколько подписчиков или несколько издателей, но разные ноды должны публиковать одинаковые темы, иначе произойдет ошибка.

1.3 Сервис (Service)

Сервисы используются для прямой коммуникации с нодами и получения обратной связи. Сервис состоит из пары сообщений, одно из которых отправляет сообщение запроса, а другое принимает и отвечает на него. Сервисы разрабатываются пользователем, хранятся в папке srv и преобразуются в исходный код путем компиляции.

1.4 Сообщение (Message)

Сообщения используются для передачи данных от одной ноды к другой путем размещения сообщения в заранее заданной топике. Все данные в ROS могут быть определены как сообщения. Сообщения отправляются с фиксированным типом и структурой: имя пакета функций/.msg имя файла.

Структура управления для построения всех процессов может быть построена с использованием нод, топиков, сервисов и сообщений. Это показано на рисунке 1.

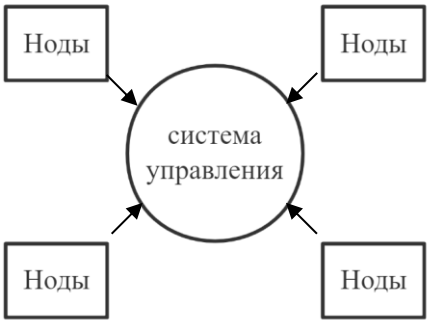


Рис. 1. Структура управления ROS

2. Описание алгоритма системы управления

Управление коллаборативными роботами требует точного контроля над суставами для достижения плавного движения и, следовательно, требует определенных алгоритмов контроля. Основным алгоритмом в данном проекте является алгоритм планирования траектории движения робота. Конструкция основана на вводе начального положения и конечного положения для управления движением робота. Исходя из этой особенности, система управления основана на алгоритме быстрого случайного дерева (Rapidly-exploring random tree – RRT) [5].

Алгоритм на основе быстро растущих случайных деревьев (Rapidly Exploring Random Tree) был изначально разработан для планирования движения неголономных механических систем в режиме реального времени. По сравнению с другими известными сэмплинг-методами, эффективность которых зависит от большого количества настраиваемых входных параметров, алгоритм RRT является наиболее универсальным средством решения широкого круга задач планирования движения.

Данный алгоритм использует в качестве дискретного представления конфигурационного пространства дерево, корень которого соответствует исходному положению объекта. Дерево достраивается таким образом, чтобы разрешение получаемой маршрутной сети увеличивалось на всем пространстве допустимых конфигураций.

Алгоритм RRT основан на случайном дереве, которое используется для обнаружения столкновений образцов точек в пространстве движения путем случайного роста узлов ветвления от корневого узла. Он не требует 3D-моделирования и решает задачу планирования пути в трехмерном пространстве и с многочисленными ограничениями. Он подходит для движения многоградусных роботов в пространстве благодаря своей высокой эффективности, точности и простоте применения.

Алгоритм RRT принимает начальную позицию в качестве корневого узла и генерирует дочерний узел методом случайной выборки, в результате чего, в свою очередь, генерируется расширенное случайным образом дерево. Когда случайно расширенное дерево касается целевого узла, в случайно расширенном дереве

находится путь от корневого узла до целевого узла, т.е. происходит планирование пути для движения робота. Основной алгоритм показан ниже:

Function RRTPlan: BOOL(*env*:environment, *T*:RRTTree, *S*goal: *node*)

1. Var *S*target, *S*nearest, *S*new, *node*
2. While(search time/space remaining) do
3. *S*target=ChooseTarget(*S*goal)
4. *S*nearest=Nearest(*T*, *S*target)
5. IF (Distance (*S*nearest, *S*goal) < Distance Threshold) then
6. Return true
7. *S*new=Extend (*S*nearest, *S*target)
8. IF (*S*new ≠ NULL) then
9. T.AddNODE(*S*new)
10. Return false

Function ChooseTarget (*S*goal: node) : ndoe

11. Var *p*: real
12. *p* = Random(0, 1.0)
13. If o<*p*<aim then
14. Return goal
15. Else if aim<*p*1<1.0 then
16. Return RandomNode ();

В процессе роста случайного дерева (строки с 1 по 10) начальным узлом является корневой узел S_{init}. Во-первых, функция ChooseTarget генерирует случайную точку S_{target} в рабочем пространстве; во-вторых, функция Nearest выбирает ближайший к S_{target} переходный узел $S_{nearest}$ в случайной функции; наконец, функция Extend увеличивает расстояние α (α - минимальный шаг алгоритма RRT) от $S_{nearest}$ до S_{target} для получения нового узла S_{new}. Если S_{new} сталкивается с препятствием, функция Extend возвращает 0, что считается неудачей; в противном случае S_{new} добавляется в случайное дерево. Повторяйте

описанные выше шаги, пока случайное дерево не вырастет до целевой точки или расстояние от целевой точки не станет меньше порогового значения, тогда алгоритм будет завершен. Пример алгоритма RRT, используемого в движении манипулятора, показан на рисунке 2.

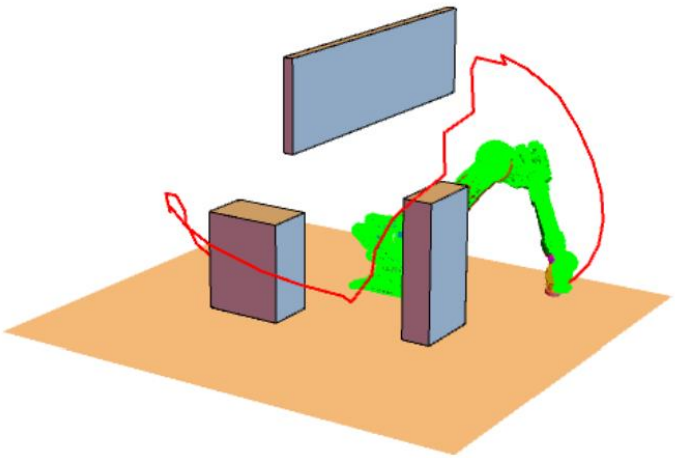


Рис. 2. Применение алгоритма RRT на манипулятор

Чтобы ускорить планирование пути, случайное дерево улучшается тем, что сначала вычисляется средняя точка начального положения и конечного положения в пространстве, и робот движется сначала к средней точке, а затем к конечной позиции, так что случайная вероятность может определить, является ли следующая точка правильной целевой точкой или случайной точкой. Параметр aim задается заранее, и генерируется случайное число *bp* от 0 до 1. Когда 0<*bp*<aim, случайное число движется к целевой точке; когда aim<*bp*<1, случайное число растет в случайном направлении [6].

Алгоритм RRT можно используется для расчета и вывода скорости вращения и радиан каждого сустава во время движения робота при вводе начального положения робота и целевого положения.

3. Проектирование коммуникационного интерфейса

Модуль коммуникационного интерфейса используется для установления соединения между главным компьютером и коллаборативным роботом. Он используется для рассылки управляющей информации и передачи обратной связи между ними. Традиционная коммуникация ROS основана на ID-соединении,

которое применимо только для нескольких продвинутых роботов, таких как UR, а коммуникация Controller Area Network (CAN) является общеприменимой на сегодняшний день [7]. Поэтому в данной работе используется только алгоритм планирования пути и среда симуляции в ROS, и сочетается с CAN-связью для создания нового метода связи.

Связь CAN используется для передачи параметров двигателя, таких как идентификатор двигателя, угол поворота, скорость и т.д., рассчитанных главным компьютером, на микросхему управления каждого двигателя робота для управления роботом. Этот метод связи не использует коммуникацию ROS и устанавливает связь с двигателями напрямую. Этот метод дополняет существующие характеристики устойчивости и оперативности, и добавляет такие характеристики, как простота связи, широкая применимость и высокая портативность, которые могут быть применены к большинству коллаборативных роботов на рынке.

4. Основной процесс управления коллаборативным роботом

Процесс управления делится на три части: операции планирования пути, коммуникационные соединения и управление двигателем. Управление коллаборативным роботом – это фактически управление углом поворота, скоростью и временем переключения двигателей на каждом суставе. Таким образом, этапы системы управления следующие:

(1) Построить 3D-обзор робота, вызвать конфигурацию среды ROS и задать начальное положение концевого эффектора.

(2) Открыть настроенного робота в среде ROS 3D визуализации (Rviz), перетащить цепь или ввести координаты конечного эффектора в интерфейсе редактирования, вызвать алгоритм RRT для расчета угла поворота, скорости, ускорения и положения, необходимых для каждого сустава для достижения целевого положения, и отправить результаты расчета по очереди.

(3) Связь между главным компьютером и коллаборативным роботом устанавливается посредством CAN-связи, и рассчитанный угол поворота каждого сустава и другие необходимые параметры отправляются на робот через интерфейс связи.

(4) Робот получает данные, преобразует их в распознаваемые двигателем параметры и посылает их на двигатель для управления движением робота.

Схема процесса управления показана на рисунке 3.

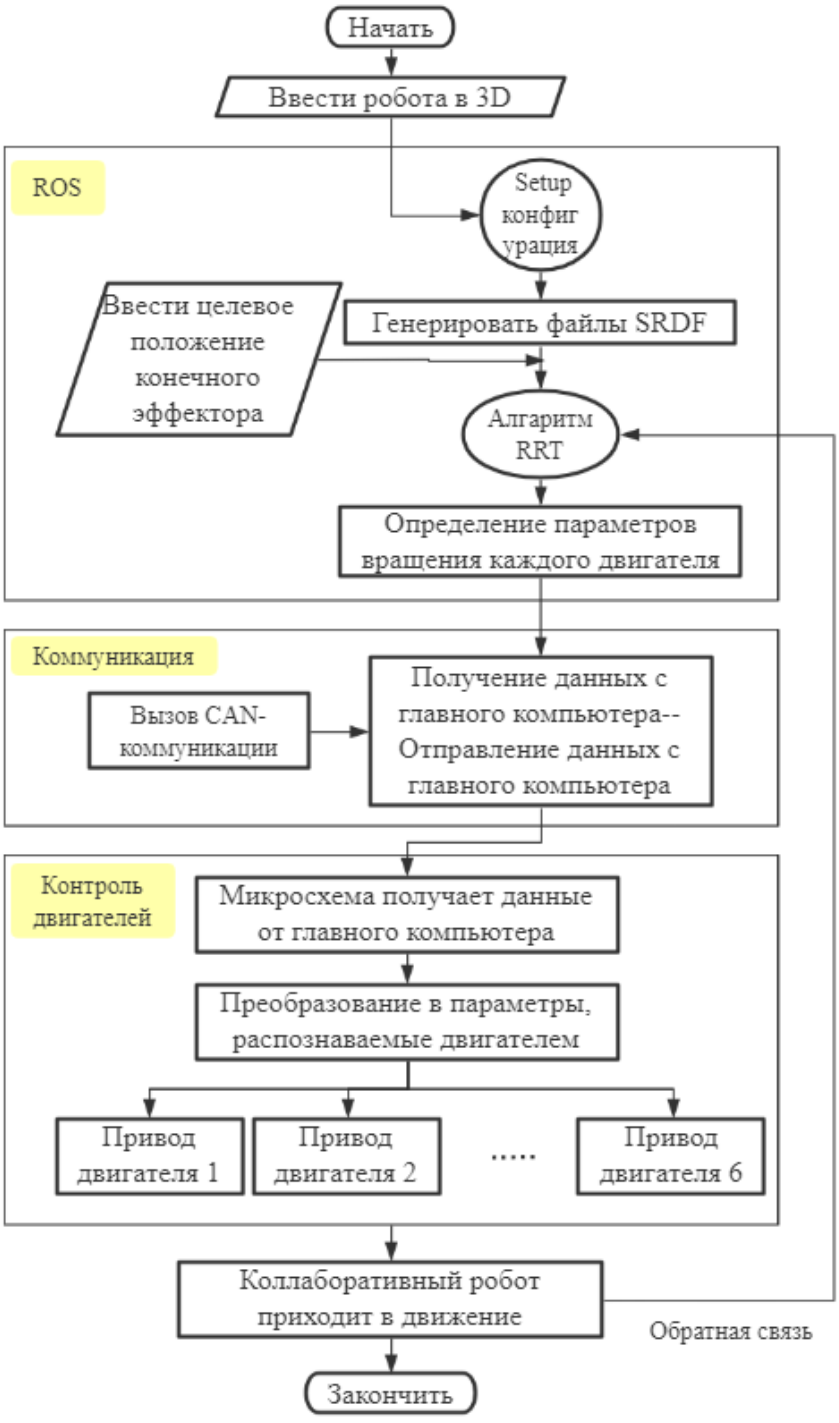


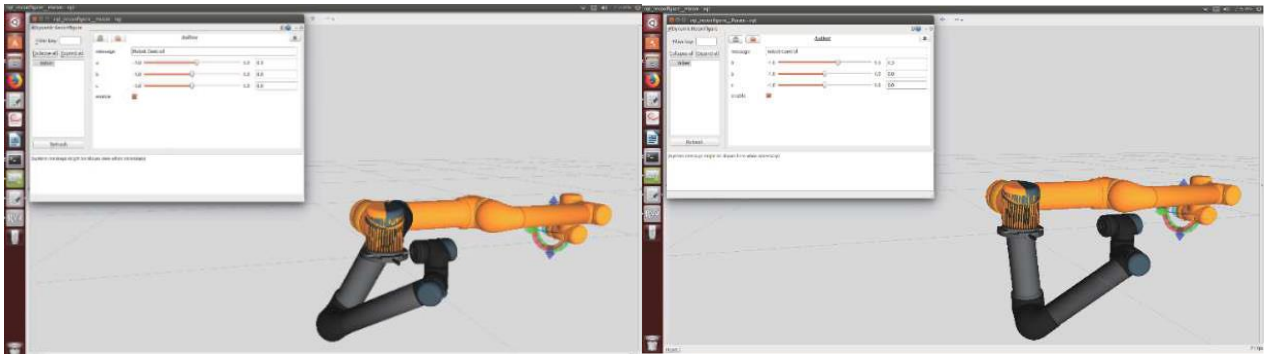
Рис. 3. Блок-схема процесса управления коллаборативным роботом

5. Результаты экспериментов и анализ

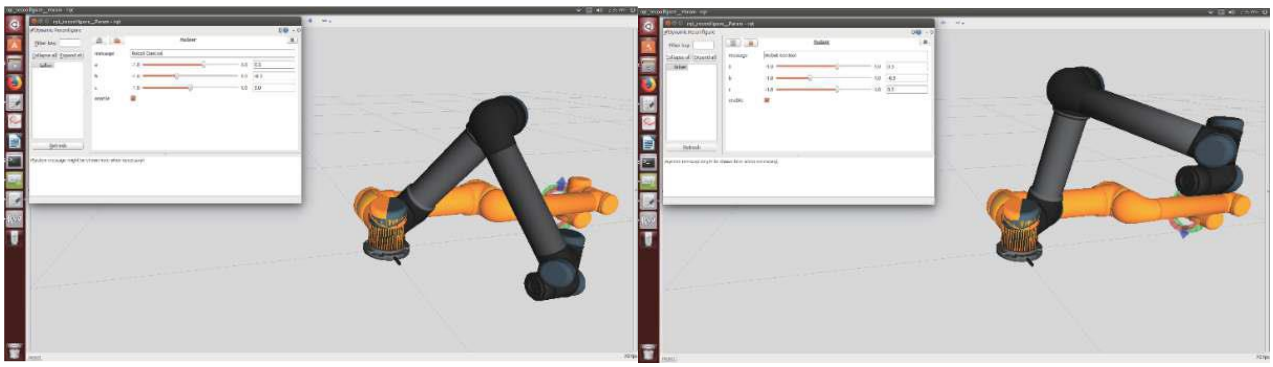
Для того чтобы проверить практическое управление системой управления коллаборативным роботом, проводятся эксперименты с коллаборативным роботом UR5. Экспериментальную платформу построили следующим образом.

5.1 Эксперимент по симуляции

В симуляции создается интерфейс управления, где можно задавать и изменять координаты конечного эффектора для управления симулируемым роботом в rviz. Начальное положение робота наблюдается как горизонтальное через интерфейс моделирования, а целевое положение может быть изменено путем перетаскивания или ввода координат концевого эффектора в интерфейсе. Во-первых, координаты конечного эффектора возвращаются в исходное положение. Во-вторых, координаты конечного эффектора изменяются в интерфейсе для того, чтобы робот двигался к целевой позиции. Задаются целевые позиции для изменения робота в каждом направлении и планируется соответствующий путь. Моделирование показано на рисунке 4.



а) Начальное положение б) Позиция 1



в) Позиция 2 г) Позиция 3

Рис. 4. Моделирование эксперимента по управлению роботом

Результаты моделирования показывают, что введя целевые координаты конечного эффектора, можно вызвать алгоритм RRT в программе моделирования для расчета угла поворота каждого сустава и отразить его в интерфейсе моделирования, тем самым реализовав управление моделируемым коллаборативным роботом. Конкретные координаты, введенные в эксперимент, приведены в таблице 1.

Таблица 1. Координаты входного конечного эффектора

Точки позиции	Указанные координаты
Начальное положение	(0, 0, 0)
1	(0.3, 0, 0)
2	(0.3, -0.3, 0)
3	(0.3, -0.3, 0.3)

5.2 Эксперимент по управлению коллаборативным роботом

В качестве физического робота используется коллаборативный робот UR [10]. Цепь протаскивается на интерфейс главного компьютера, и информация передается на микросхему, расположенную в роботе. Микросхема анализирует полученные данные и преобразует их в параметры, которые могут быть распознаны двигателем, тем самым управляя скоростью и временем вращения двигателя. Эксперимент по роботу показан на рисунке 5. После отправки команды с головного компьютера, контроллер двигателя получает и выполняет соответствующее действие, завершает планирование пути, и робот выполняет соответствующее действие.



Рис. 5. Эксперименты по управлению физическим роботом

Из-за ошибок в точности аппаратного обеспечения робота и отклонений в алгоритме ROS, отправленные параметры движения были откалиброваны в системе управления. Робот приводился в движение с постоянной скоростью, при этом абсолютные положения шарниров считывались, в результате видно, что робот не перемещался в заданное положение в пределах 2 мм. После изменения параметров управления и доработки программы пространственное движение робота можно было точно контролировать через интерфейс главного компьютера, что обеспечивало высокую надежность и производительность в реальном времени.

Вызвав функцию печати ROS, скорость, время и ускорение каждого сустава можно вывести на экран вывода. В таблице 2 указано время, требуемое для перемещения робота в заданные позиции после ввода координат конечного эффектора из таблицы 1. Время выхода показывает, что движение происходит мягко, с хорошей устойчивостью и производительностью в реальном времени.

Таблица 2. Время, требуемое для перемещения робота в заданные позиции

Точки позиции	Время
Начальное положение	0
1	1.94
2	1.89
3	2.02

6. Заключение

В данной статье управление коллаборативным роботом реализовано на Ubuntu с помощью ROS. Алгоритм планирования пути был оптимизирован, а коммуникация между главным компьютерами и контроллером двигатели в ROS была усовершенствована, чтобы создать метод коммуникации, который применим и прост в эксплуатации, сохраняя при этом оригинальное реальное время и надежность. В результате экспериментов очевидно, что система управления может реализовать основную работу коллаборативного робота. Это имеет прикладное значение. Однако, развитие других функций, таких как разработка системы технического зрения и разработка концевых захватов, требует дальнейшего изучения для достижения лучших результатов и перспектив.

Литература

1. Командная разработка с использованием Visual Studio Team Foundation Server [Electronic resource]. – URL: https://visualstudio.microsoft.com/
2. Дьяконов В. П. MATLAB. Полный самоучитель. – М.: ДМК Пресс, 2012.
3. Martinez A. Learning ROS for Robotics Programming[M]. Packt Publishing, 2013.
4. Hart S, Dinh P, Hambuchen K. The Affordance template ROS package for robot task programming. Proceedings of the IEEE International Conference on Robotics and Automation. Seattle, WA, USA. 2015. 6227-6234.
5. Martin SR, Wright SE, Sheppard JW. Offline and online evolutionary bi-directional RRT algorithms for efficient replanning in dynamic environments. Proceedings of IEEE International Conference on Automation Science and Engineering. Scottsdale, AZ, USA. 2007. 1131-1136.
6. S. Li, D. Zhao, Y. Sun, "Path Planning Algorithm Based on the Improved RRT-Connect for Home Service Robot Arms," 2021 IEEE International Conference on Intelligence and Safety for Robotics (ISR), 2021, pp. 403-407, doi: 10.1109/ISR50024.2021.9419385.
7. A. Ghatak, A. A. Kumar, "Controller Area Network Bus based Communication system for Thermal Power Plant," 2020 Third International Conference on Advances

in Electronics, Computers and Communications (ICAECC), 2020, pp. 1-6, doi: 10.1109/ICAECC50550.2020.9339516.

References

1. Team Development Using Visual Studio Team Foundation Server [Electronic resource]. – URL: https://visualstudio.microsoft.com/
2. Dyakonov V. P. MATLAB. Complete Tutorial. – M.: ДМК Пресс, 2012.
3. Martinez A. Learning ROS for Robotics Programming[M]. Packt Publishing, 2013.
4. Hart S, Dinh P, Hambuchen K. The Affordance template ROS package for robot task programming. Proceedings of the IEEE International Conference on Robotics and Automation. Seattle, WA, USA. 2015. 6227-6234.
5. Martin SR, Wright SE, Sheppard JW. Offline and online evolutionary bi-directional RRT algorithms for efficient replanning in dynamic environments. Proceedings of IEEE International Conference on Automation Science and Engineering. Scottsdale, AZ, USA. 2007. 1131-1136.
6. S. Li, D. Zhao, Y. Sun, "Path Planning Algorithm Based on the Improved RRT-Connect for Home Service Robot Arms," 2021 IEEE International Conference on Intelligence and Safety for Robotics (ISR), 2021, pp. 403-407, doi: 10.1109/ISR50024.2021.9419385.
7. A. Ghatak, A. A. Kumar, "Controller Area Network Bus based Communication system for Thermal Power Plant," 2020 Third International Conference on Advances in Electronics, Computers and Communications (ICAECC), 2020, pp. 1-6, doi: 10.1109/ICAECC50550.2020.9339516.

© Сун Бисянь, Ван Вэньюэ, 2022 Научно-образовательный журнал для студентов и преподавателей «StudNet» №5/2022.

Для цитирования: Сун Бисянь, Ван Вэньюэ. СИСТЕМА УПРАВЛЕНИЯ КОЛЛАБОРАТИВНЫМ РОБОТОМ НА ОСНОВЕ ROS // Научно-образовательный журнал для студентов и преподавателей «StudNet» №5/2022.