Санкт-Петербургский государственный университет

Кафедра механики управляемого движения

Кузнецов Александр Петрович

Выпускная квалификационная работа бакалавра

Построение траекторий трёхмерного движения антропоморфного механизма

Направление 010300

Фундаментальная информатика и информационные технологии

Научный руководитель,
кандидат физ.-мат. наук,
доцент
Латыпов В. Н.

Санкт-Петербург

2017

Содержание

Введение

Разработка аппаратов, перемещающихся с помощью конечностей, стала одним из важных направлений робототехники. Перемещаться по поверхности с неровностями целесообразнее посредством шагания: шагающие объекты используют лишь участки местности, необходимые для постановки ног; благодаря этому, требования к поверхности передвижения снижаются.

Ряд исследований посвящен теоретическим вопросам, а также вопросам конструирования и лабораторного макетирования многоногих шагающих механизмов. Управление ходьбой таких устройств осуществляется на кинематическом уровне - локомоции организуются с помощью последовательности статически устойчивых конфигураций. С увеличением числа ног шагающих роботов проблема структуры управления упрощается; с другой стороны, вследствие растущего числа степеней свободы, их механическая часть становится более сложной.

В случае наличия на пути робота препятствий, задача построения траектории движения механизма значительно усложняется. В этом случае траектории требуется вычислять, основываясь не только на параметрах самого механизма, но и на расположении, форме и состоянии объектов в окружающей среде робота.

В данной работе рассматривается задача построения трехмерной траектории движения двуногого механизма в пространстве с препятствиями. По известной карте местности, определяемой набором прямоугольных препятствий, и заданным начальной и конечной точках искомого маршрута вычисляется последовательность положений механизма в пространстве. В каждый момент времени полный набор обобщенных координат робота вычисляется по положениям ступней.

Для решения задачи применяется алгоритм быстрорасширяющихся деревьев (Rapidly expanding Random Trees, RRT). Рассмотрен способ

ускорения работы алгоритма RRT с помощью построения дополнительного поискового k-мерного дерева.

Постановка задачи

Рассматривается антропоморфный механизм в аксиальной плоскости. О разумности такого подхода говорится в работе [13]. Для описания конфигурации стоп (Рис. 1) механизма введем обобщенные координаты $q = [x_1, y_1, x_2, y_2, \varphi]$, в инерциальной системе координат XY (ось X направлена по горизонтали, ось Y направлена вертикально вверх). Обозначим x_1, y_1 - координаты левого нижнего угла левой стопы, x_2, y_2 - координаты левого нижнего угла правой стопы, за φ примем угол левой стороны любой стопы относительно оси Ox. Угол является положительным, если стопа отклоняется от горизонтали в направлении, противоположном ходу часовой стрелки. Используется один угол, так как для описании поворота требуется несколько конфигураций с разными углами (см. параграф 2.6). d - длина шага при ходьбе вдоль прямой, h - длина стопы, w - ширина стопы. d, h, w известны заранее.

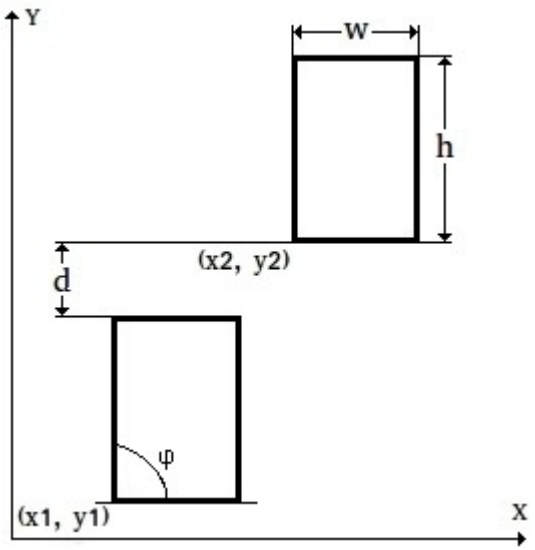

Рис. 1. Конфигурация из множества C

Механизм либо перемещается по прямой, либо поворачивается. Каждая их этих фаз накладывает ограничения на пересчет координат стоп.

При движении вдоль прямой, координата φ не изменяется.

Задача автоматического построения карты местности по данным различных сенсоров (видеокамер, лазерных дальномеров и т.д.) сама по

себе очень трудна и в данной работе не рассматривается. Пусть посредством какого-либо сервиса робот получает карту местности. Карта представлена набором препятствий в виде прямоугольников, заданных координатами нижнего левого и верхнего правого углов. В том случае, если препятствия могут изменять свое положение, карта должна обновляться. Также заданы координаты начального и конечного положений стоп робота. На Рис. 2 приведен пример карты местности с отмеченными начальным и конечным положениями робота.

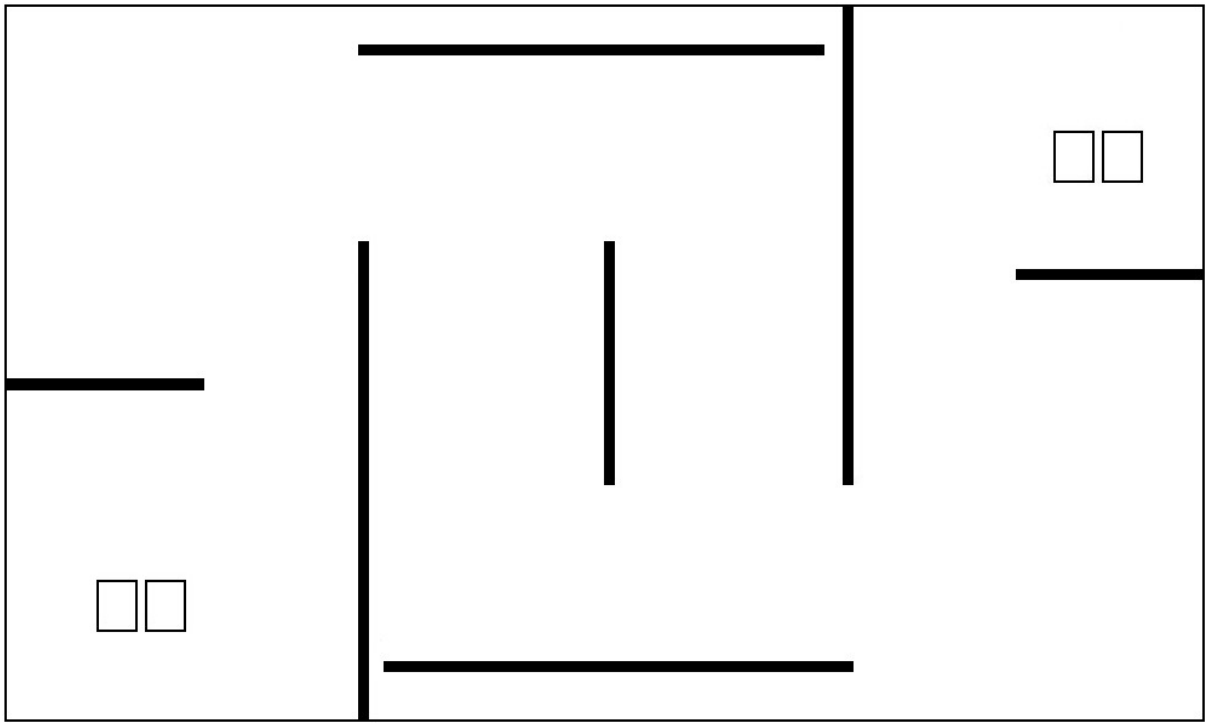

Рис. 2. Графическая формулировка задачи

Задача заключается в построении трехмерной траектории перемещения робота из начального положения в конечное, которое может быть представлено положением его стоп и центра масс. Также должны быть заданы длина бедра, длина голени и высота таза — эти величины необходимы для расчета положения робота (рассматривается в параграфе 2.8). Траектория должна быть представлена дискретным набором состояний, в которые робот способен попасть с учетом его динамики и препятствий, представленных на карте местности.

Обзор литературы

В работах А.М. Формальского [1] и В. В. Белецкого [3] подробно рассмотрена динамика двуногих механизмов и способы получения уравнений движения. В [1] рассмотрены вопросы получения уравнений движения, а в [3] рассматриваются способы качественного исследования решений сведением полученных уравнений к движению относительно центра масс.

Поиск пути в двухмерном и трехмерном пространствах рассматривается в трудах [5], [6], [7], [9], [10]. Обход препятствий сводится к построению траекторий положения стоп с помощью вероятностных алгоритмов (Probabilistic roadmap, potential fields). Плюсы методов такого типа в том, что они могут учитывать неровности поверхности, а их скорость работы позволяет использовать поиск пути неограниченное количество раз, уменьшая область, что может способствовать обходу движущихся препятствий.

Большой вклад в планирование движения внес Стивен Лавалль с публикациями алгоритма [4], [8], [11], основанном на заполнении пространства быстро расширяющимися деревьями. Его работы уже нашли применение в планировании пути автомобилей и летающих объектов: вертолетов, мультикоптеров.

Для оптимизации последнего в данной работе используется алгоритм KD-tree, описанный в работе [12].

В следующих главах будут подробнее рассмотрены оптимизация RRT средством построения KD-tree, а также представлено применение полученного алгоритма для построения траекторий движения робота.

Глава 1. Алгоритм построения траектории RRT

1.1 RRT

Планирование траектории рассматривается как поиск в непрерывном пространстве конфигураций C, где $q \in C$ определяет положение робота в окружающем пространстве. C_{free} – подпространство пространства состояний, в котором робот не сталкивается ни с какими препятствиями.

Быстро исследующие случайные деревья [8] – динамическая структура данных, предназначенная для исследования пространства состояний. На Рис. 3 продемонстрированы этапы построения RRT.

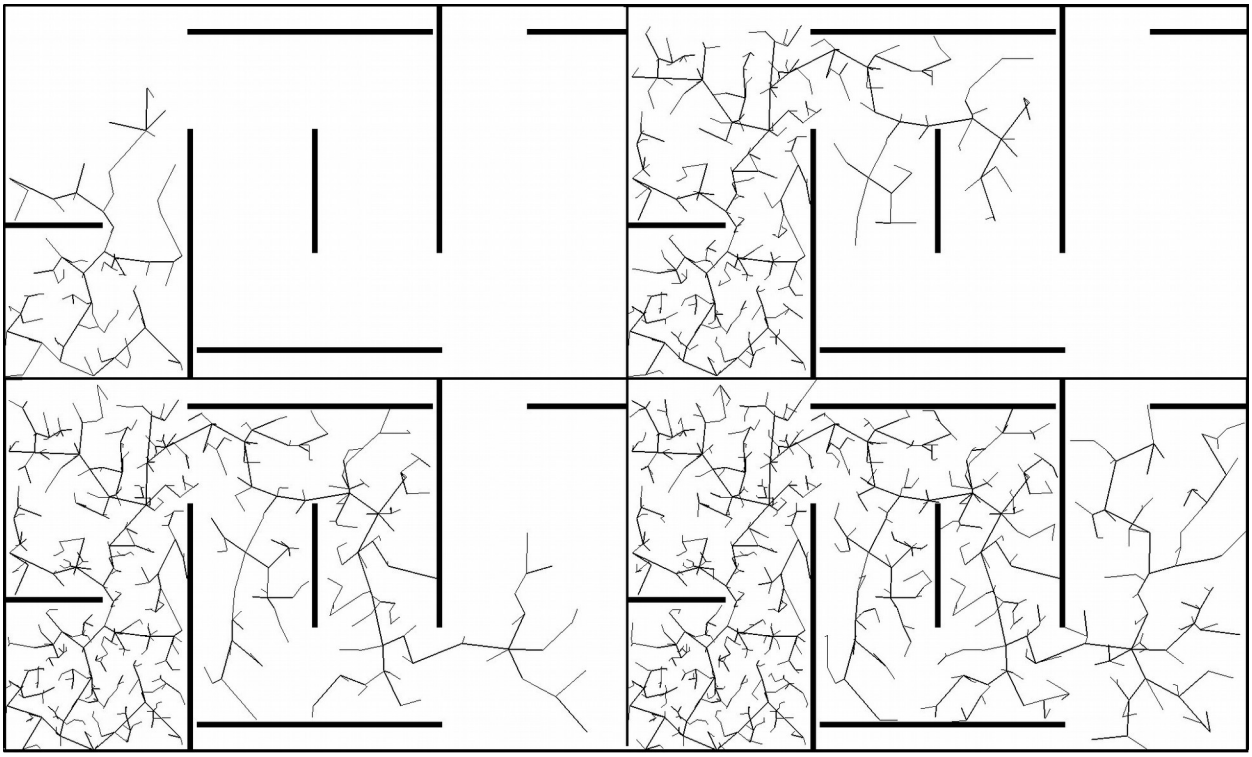

Рис.3. Некоторые шаги построения RRT

Строится дерево $G = (V, E)$, выходящее из заданной начальной точки карты и заполняющее пространство в обход препятствий. Каждая вершина из множества V принадлежит подмножеству конфигурационного пространства C. Элемент E — прямолинейный отрезок в конфигурационном пространстве, такой, что каждая точка этого отрезка представляет собой допустимую конфигурацию робота.

Псевдокод алгоритма RRT:

Search:

V.*add*(q_{start})

while (!*is_available*(q_{finish}))

q_{new} = q_{random}

expand(q_{new})

return G

Expand:

$q_{nearest}$ = *get_nearest*(V)

if (*is_available*($q_{nearest}$, q_{new}))

V.add(q_{new})

E.add($q_{nearest}$, q_{new})

Построение дерева начинается в процедуре *search* путем добавления стартовой конфигурации q_{start}. Далее, пока не существует пути до конечной конфигурации q_{finish} (существование пути проверяется в функции *is_available*()), генерируется случайная конфигурация q_{random}.

Генерация q_{random} производится с учетом карты местности: генерируемая конфигурация не должна никаким образом выходить за пределы карты и не должна иметь пересечений с препятствиями. Далее управление передается функции *expand*, аргументом которой является вновь сгенерированная конфигурация q_{new}.

В функции *expand* производится поиск по вершинам дерева с целью найти ближайшую вершину $q_{nearest}$ к сгенерированной q_{new}. Далее в функции *is_available()* исследуется достижимость вершины q_{new} из вершины $q_{nearest}$ (достижимость включает в себя заранее обусловленное расстояние и отсутствие препятствий на пути к этой вершине). Если условие достижимости удовлетворяется, то в множество вершин дерева *G*

добавляется еще одна вершина, а в множество ребер - ребро, связывающее новую вершину с ближайшей к ней вершиной дерева.

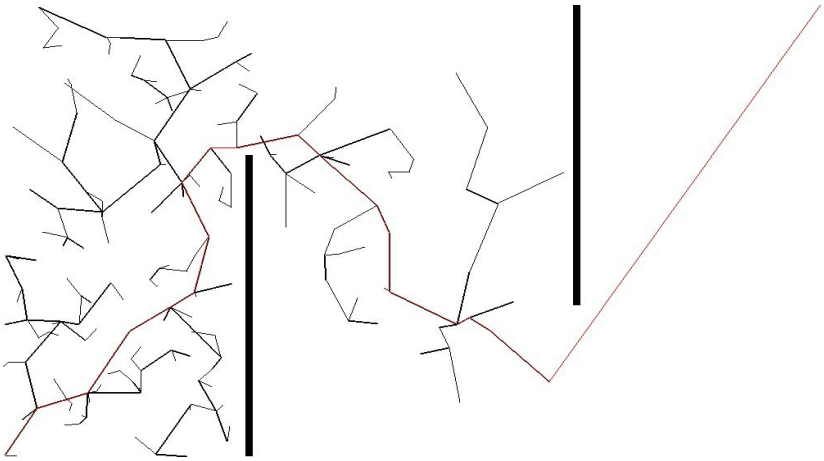

Рис. 4. Путь для точки, построенный алгоритмом RRT

Алгоритм исследует пространство состояний, начиная со стартовой конфигурации, пока не достигнет финишной конфигурации.

На Рис. 4 изображено дерево поиска пути для точки.

Самой трудоемкой частью является поиск вершины дерева, ближайшей к только что сгенерированной *get_nearest*. Линейный поиск по такому дереву осуществляется за O(n) операций, где n - количество узлов. В силу этого предлагается использовать другой способ поиска в многомерном пространстве, приведенный в следующем пункте.

1.2 KD-tree

Методы поиска в пространстве состояний [4], [5] делятся на неинформированные, которые не используют никакой информации о конкретной задаче, кроме информации о том, как отличить целевое состояние от любого другого, и информированные, которые используют дополнительную информацию о задаче, чтобы сократить перебор путем отбрасывания заведомо неподходящих вариантов.

Для ускорения поиска ближайшей точки в пространстве состояний робота к вновь сгенерированной удобно дополнительно использовать k-мерное дерево поиска (KD-tree).

k-мерное дерево - это разновидность двоичных деревьев быстрого поиска. Вообще говоря, k-мерное дерево разбивает пространство для упорядочивания своих узлов в k-мерном пространстве. Такие деревья являются несбалансированными, они делятся на однородные(каждый узел хранит состояние) и неоднородные (каждый внутренний узел хранит ключ для поиска, а лист - состояние).

Рассмотрим построение однородного K-мерного дерева для двумерного пространства (Рис. 5). Пусть заданы точки (2, 3), (5, 4), (9, 6), (4, 7), (8, 1), (7, 2). Строится дерево, рекурсивно разбивая плоскость прямыми $x = k$, $y = k$ поочередно. За k принимается соответствующая координата x или y точек.

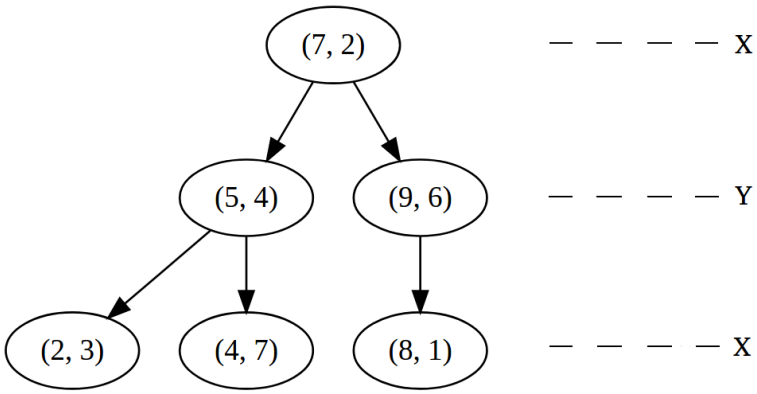

Рис. 5. KD-tree

Глава 2. Реализация алгоритма

2.1 Начальные данные

Входными параметрами алгоритма являются начальное положение стоп механизма ($q_{start} \in C$), карта местности, конечное положение ($q_{finish} \in C$). Карта представляет собой набор препятствий, к которым должно быть соответствующее геометрическое описание. По таким начальным данным можно определить положение стоп механизма на пути к конечному положению, а по описанным геометрически положениям стоп определяется положение всего механизма. Алгоритмы RRT и KD-tree используются для построения положений стоп по всей траектории от стартового положения до конечного. При наличии на карте других движущихся объектов (препятствий) необходимо использовать алгоритм чаще или ограничиваться меньшим размером карты с более частыми обновлениями.

2.2 Генерация очередного положения стоп

Скорость и результат работы алгоритма напрямую зависит от выбора конфигурационного пространства. Для построения траектории рассматриваемого механизма, подразумевается отсутствие столкновений с препятствиями. Учитывая то, что поверхность, по которой передвигается робот не содержит неровностей, для построения такого пути достаточно генерировать положения стоп механизма. Принимая во внимание тот факт, что длина шага робота ограничена, обе стопы можно отграничить прямоугольником достаточной длины (удвоенная сумма длины стопы и длины шага) и ширины (сумма удвоенной ширины каждой стопы и расстояния, опционально зависящего от механизма).

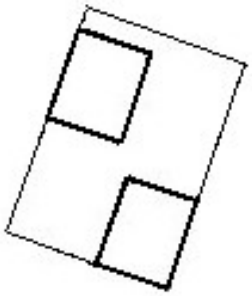

Рис. 6. Ограничивающий контур

Так как длина и ширина такого контура (Рис. 6) заранее известна, то для генерации прямоугольника достаточной будет генерация, скажем, координат левого нижнего угла и угла наклона, одной из граней к оси Ox. Грань, относительно которой генерируется угол должна быть выбрана заранее. Грань контура, сопряженную с точкой, положение которой генерируется алгоритмом и прилегающую к нижней грани стопы далее будем называть нижней гранью контура.

При повороте обе стопы стоят пятками на нижней грани контура (Рис. 7).

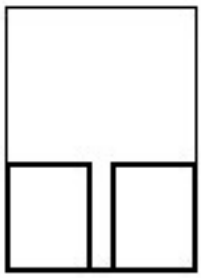

Рис. 7. Положение стоп при повороте

Далее необходимо определить допустимость вновь сгенерированной конфигурации на данной карте. Другими словами, принадлежность признакам, которым должен удовлетворять только что сгенерированный прямоугольник. Такой прямоугольник не должен:

- своими гранями выходить за пределы карты
- не должен пересекать, совпадать или содержать обозначенные на карте препятствия.

Удовлетворяющая этим требованиям конфигурация может быть рассмотрена при добавлении в деревья.

2.3 Добавление вершин в k-мерное дерево

После того, как конфигурация сгенерирована, ищется ближайшая к ней вершина дерева. Так как поиск такой вершины в RRT занимает $O(n)$, где n - количество уже принадлежащих дереву конфигураций, решено использовать однородное К-мерное дерево, как вспомогательное.

Поскольку конфигурация задается тремя координатами: координаты левого нижнего угла и угол отклонения одной из сторон от оси Ox, дерево будет делить пространство по каждой из трех координат рекурсивно в порядке x, y, φ.

По такому дереву ищется максимально близкое к вновь сгенерированному состояние, и при соблюдении описанных далее условий добавляется в это дерево согласно описанному выше алгоритму. Это есть первый этап добавления вершины в быстрорасширяющееся дерево.

2.4 Добавление вершин в RRT

Когда найдена максимально подходящая по расстоянию конфигурация из дерева алгоритм переходит ко второму этапу добавления - проверке на достижимость.

Как говорилось ранее, карта хранит информацию о препятствиях на пути механизма.

Псевдокод базовой процедуры проверки достижимости:

$is\_available\ (q_{fisrt},\ q_{second})$:

$q_{fisrt\_redirected} = redirect(q_{fisrt})$

if $(!is\_valid(q_{fisrt\_redirected}))$

return *false*

if$(!check_slice(q_{fisrt},\ q_{redirected}))$

return *false*

$q_{fisrt\_redirected\_and\_shifted} = shift(q_{first\_redirected})$

if (!*is valid*($q_{fisrt\_redirected\_and\_shifted}$))

return *false*

if(!*check_brick*($q_{first\_redirected}$, $q_{fisrt\_redirected\_and\_shifted}$))

return *false*

return *true*

На вход процедуре *is_available* подаются две конфигурации q_{fisrt}, q_{second}. Необходимо проверить наличие прямолинейного пути из одной конфигурации до другой. Сначала создается дополнительная конфигурация $q_{fisrt\_redirected}$ (Рис. 8),

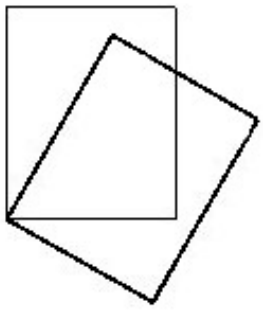

Рис 8. Дополнительная конфигурация $q_{first\_redirected}$ (снизу справа)

координаты которой отличаются от координат первой входной конфигурации q_{fisrt} лишь углом.

Тем самым получена конфигурация, повернутая по направлению к q_{second}. Далее проверяется допустимость полученной конфигурации для данной карты *is_valid*(). В качестве проверки на пересечение конфигураций с препятствиями можно использовать алгоритм, основанный на векторном произведении, приведенный в книге [14].

Если конфигурация допустима, проверяется возможность такого поворота процедурой *check_slice*().

Если поворот возможен и полученная после поворота конфигурация допустима, проверяется конфигурация $q_{redirected\_and\_shifted}$, полученная параллельным переносом конфигурации $q_{redirected}$ на позицию q_{second}.

Если и такая конфигурация допустима, посредством функции *check_brick*() проверяется возможность вышеупомянутого параллельного переноса $q_{redirected}$ на позицию $q_{redirected\_and\_shifted}$ (Рис. 9).

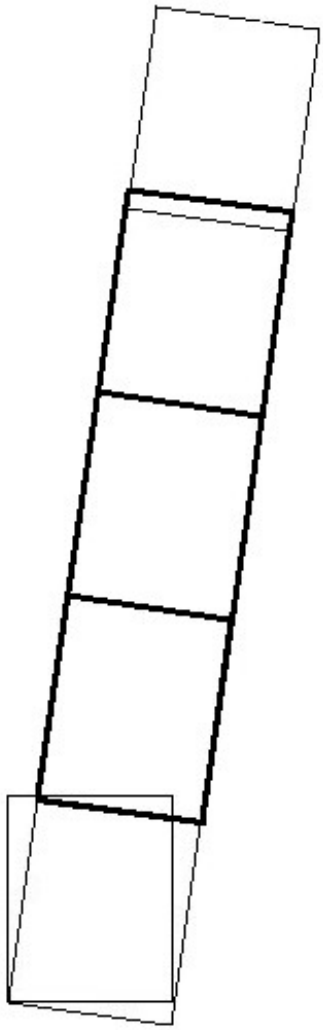

Рис. 9. Конфигурации для проверки *check_brick()* (посередине). Конфигурация $q_{redirected\_and\_shifted}$ (справа сверху).

Если ни один этап проверки не вернул *false*, то функция возвращает *true*, и завершает свою работу.

2.5 Оптимизация пути

Путь получается добавлением к вершинам RRT дополнительных конфигураций для разворота и последовательного перемещения.

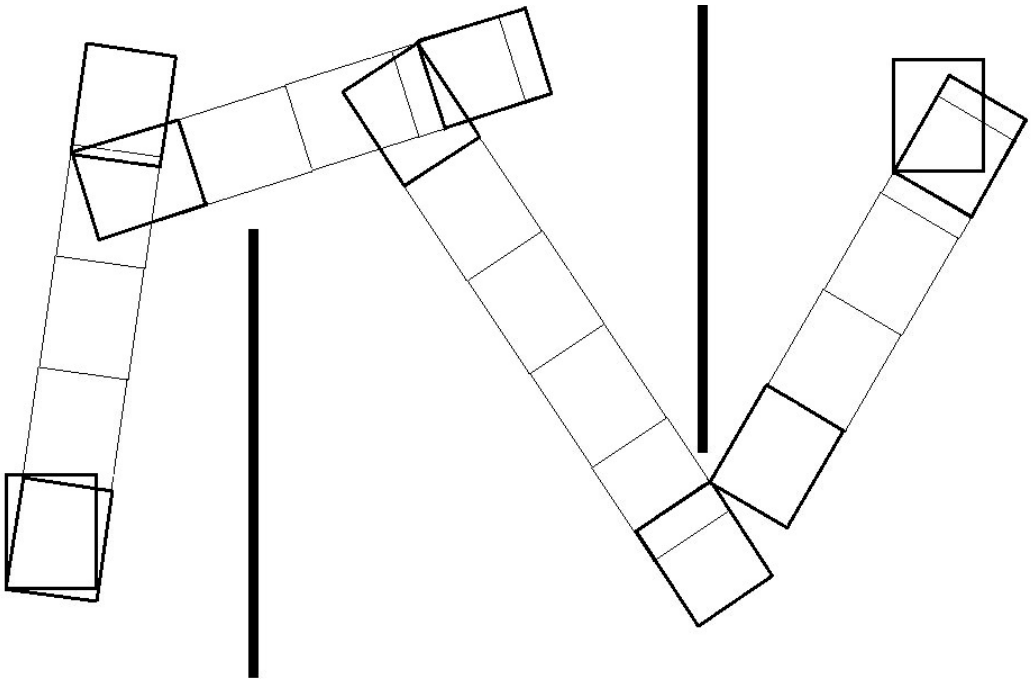

Рис. 10. Оптимизированный путь из ограничивающих контуров

Алгоритм возвращает неоптимальную траекторию, нуждающуюся в дополнительной оптимизации. Для этой цели подходит алгоритм вырезания углов. На траектории выбирается три смежных узла. Если перемещение по прямой из крайнего левого узла в крайний правый возможно, то средний узел удаляется, при этом первому конфигурации узла, из которого двигается механизм, присваивается новый угол - направление к правому узлу. Процесс повторяется итеративно, пока не будут удалены все возможные узлы. Оптимизированный путь показан на Рис. 10.

2.6 Расчет положений стоп

Как было указано, стопы ограничены прямоугольным контуром, их положение при движении вдоль прямой фиксировано. На Рис. 11 продемонстрирована последовательность стоп робота при начале ходьбы с левой ноги.

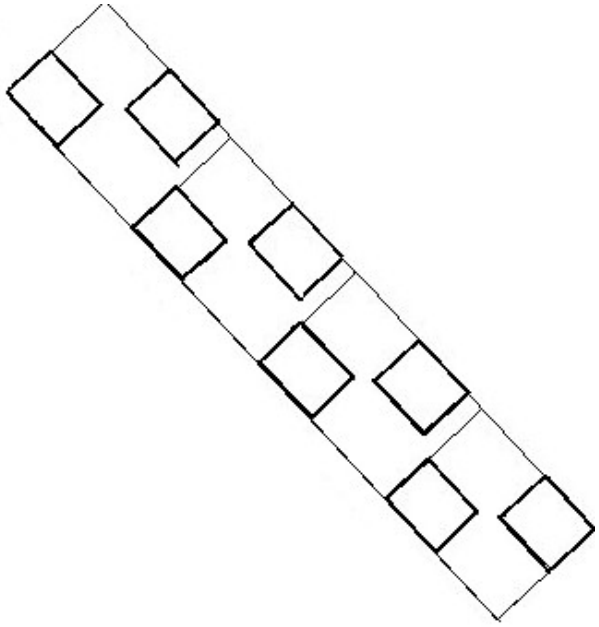

Рис.11. Расположение стоп робота при движении вдоль прямой линии

Но для поворота это не подходит, когда при построении пути достигается состояние поворота (две конфигурации в пути идут с одинаковыми координатами левого нижнего угла ограничивающего прямоугольника), правая нижняя стопа (или левая, если она выше) смещается вниз, на нижнюю сторону контура. Так получаются конфигурации для разворота робота (робот поворачивается приставными шагами).

2.7. Расчет позы робота

По двум последовательным положениям стоп определяются необходимые промежуточные положения, по которым вычисляются все остальные обобщенные координаты механизма для всех моментов времени. Один из возможных алгоритмов построения траекторий всего механизма приведен в [15].

Конфигурация всего механизма задается обобщенными координатами $[x_0, y_0, z_0, \alpha_{left}, \alpha_{right}, \theta_{left}, \psi_{left}, \gamma_{left}, \theta_{right}, \psi_{right}, \gamma_{right}, \theta_{torso}, \psi_{torso}, \gamma_{torso}]$, где x_0, y_0, z_0 — центр тазобедренного сустава, $\alpha_{left}, \alpha_{right}$ - углы крепления голени к стопам, $\theta_{left}, \psi_{left}, \gamma_{left}, \theta_{right}, \psi_{right}, \gamma_{right}$ — углы, задающие положения

левого и правого бедра соответственно, θ_{torso}, ψ_{torso}, γ_{torso} — углы, задающие положение торса в пространстве.

Углы наклона туловища вычисляются исходя из глобальных задач робота и дополнительных ограничений. Поэтому для построения позы робота по положениям стоп достаточно рассчитать положение бедра и угол наклона голени к плоскости стопы (Рис. 12).

Пусть точка крепления бедра к тазу имеет координаты (x, y, z), а точка крепления голени к стопе (x_1, y_1, z_1). Положение бедра задается тремя углами (θ, ψ, $\gamma = 90$), положение голени задается углом α.

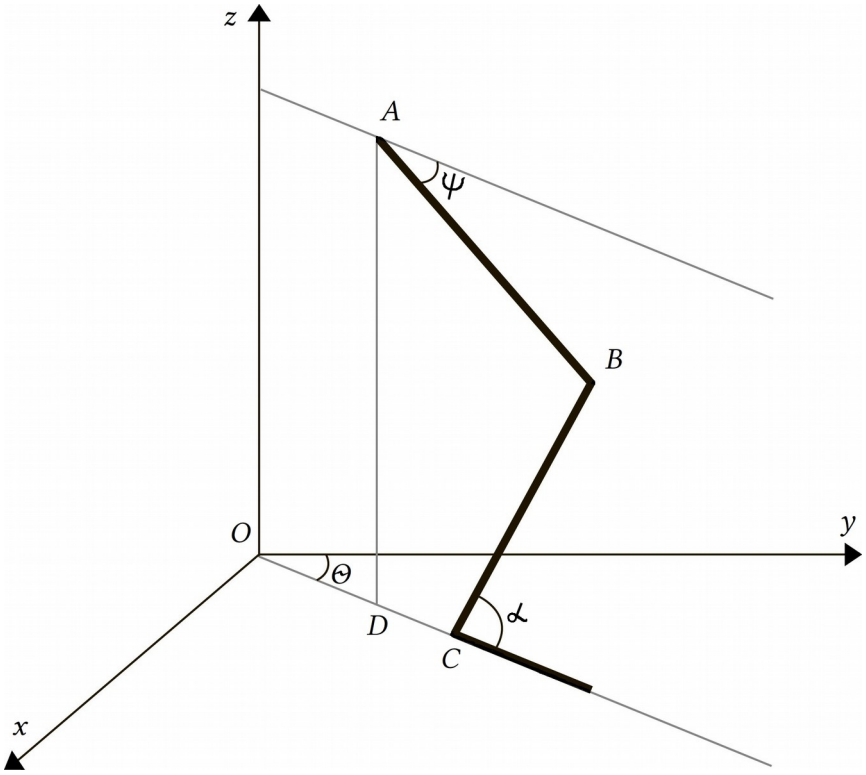

Рис. 12. Нога механизма (при стопе впереди)

Обозначив длину бедра l_1, а длину голени l_2, для вычисления положения ноги, стопа которой при шаге впереди, имеем формулы:

$$DC = \left|\sqrt{x_1^2 + y_1^2} - \sqrt{x^2 + y^2}\right| \; ; \; AC = \sqrt{z^2 + DC^2} \; ;$$

$$P_{ABC} = AC + l_1 + l_2 \; ; \; S_{ABC} = \sqrt{\frac{P_{ABC}}{2}\left(\frac{P_{ABC}}{2} - AC\right)\left(\frac{P_{ABC}}{2} - l_1\right)\left(\frac{P_{ABC}}{2} - l_2\right)} \; ;$$

$$\alpha = 180 - \arcsin\left(\frac{z}{DC}\right) - \arcsin\left(\frac{2 * S_{ABC}}{(AC * l_2)}\right) \; ;$$

$$\theta = \quad 90 - \arcsin\left(\frac{DC}{AC}\right) - \arcsin\left(\frac{2 * S_{ABC}}{AC * l_2}\right) \quad ;$$

$$\psi = \quad \arccos\left(\frac{y}{\sqrt{x^2 + y^2}}\right) \quad .$$

Аналогичные формулы получаются и для ноги, стопа которой при шаге сзади.

Глава 3. Эксперименты

Для проверки алгоритма разработана программа, принимающая на вход параметры робота, карту местности и координаты стоп конечной и начальной конфигураций.

3.1 Описание работы расчетной программы

После ввода данных запускается поиск промежуточных позиций стоп с учетом карты местности — алгоритм RRT. Вершины дерева хранятся в массиве. Каждая вершина из множества V представляется соответствующим набором координат стоп и массивом индексов массива вершин дерева, соответствующим детям этой вершины. При запуске алгоритма, в массив вершин V RRT добавляется вершина, получаемая из начальной конфигурации механизма q_{start}, путем перевода координат из обобщенных в генерируемые, массив детей пуст. Эта же вершина является корнем вспомогательного k-мерного дерева. В процессе выполнения алгоритма вычисляется q_{random}, из которой после проверки на соответствие карте получается $q_{new} \in C$, находится ближайшая в ней вершина $q_{nearest} \in V$ с помощью поиска по k-мерному дереву. Затем прямолинейный путь (q_{new}, $q_{nearest}$) проходит проверку на отсутствие пересечений с препятствиями. В случае успешного выполнения этих условий q_{new} добавляется в массив вершин RRT, а в массив детей $q_{nearest}$ индекс добавленной вершины. В k-мерное дерево добавляется эта же вершина по правилам добавления, описанным в параграфе 2.3. Если из последней добавленной вершины RRT достижима конечная конфигурация $q_{finish,}$ то путь найден.

Набор узлов RRT, приводящих из стартового положения стоп q_{start} к конечному q_{finish} составляют искомый путь. Далее путь оптимизируется (параграф 2.5), происходит расчет положений стоп механизма (параграф 2.6) и промежуточных поз робота (параграф 2.7).

3.2 Пример работы расчетной программы

В расчетную программу подается описание механизма из таблицы 1.

Параметр	Значение
Длина бедра	120
Длина голени	110
Высота таза	225
Ширина таза	80
Длина стопы	50
Ширина стопы	35
Длина шага	15

Таблица 1. Характеристики механизма

На Рис. 13-16 представлены различные результаты расчетной программы для заданных карт местности с одинаковой шириной (600) и высотой (1000) и разным количеством препятствий.

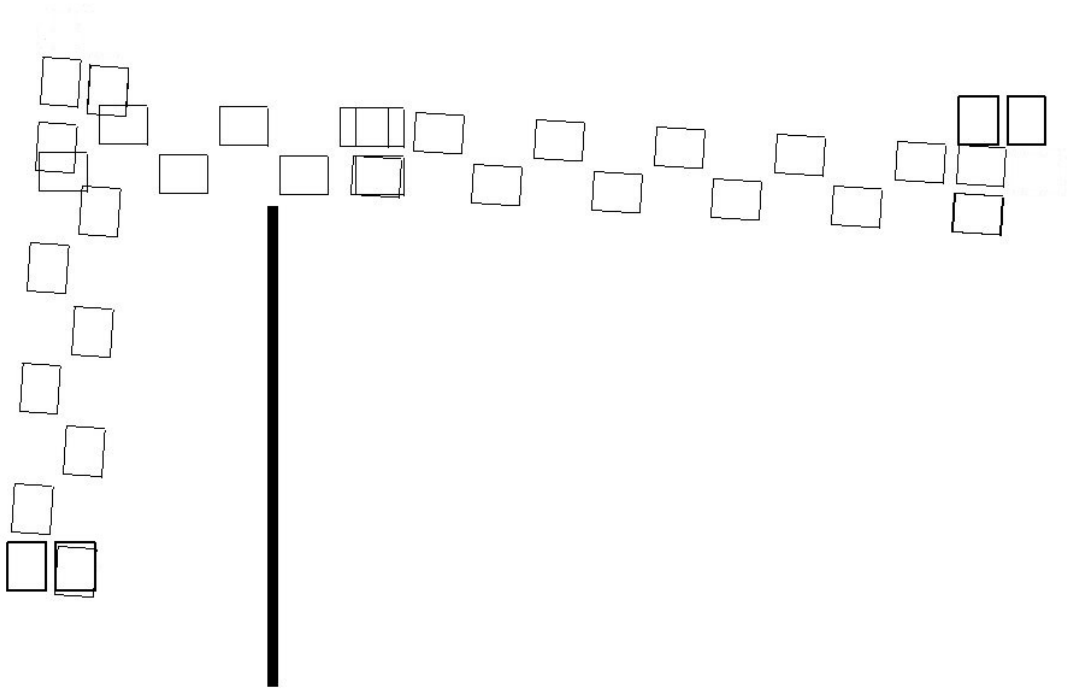

Рис. 13. Одно препятствие

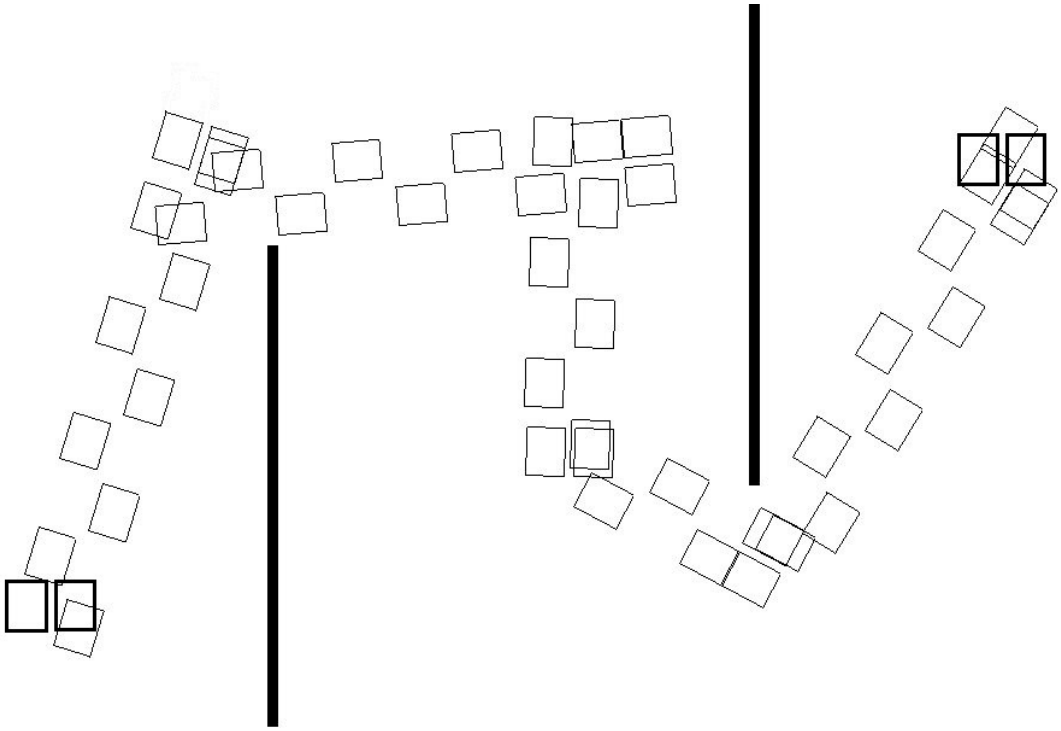

Рис. 14. Два препятствия

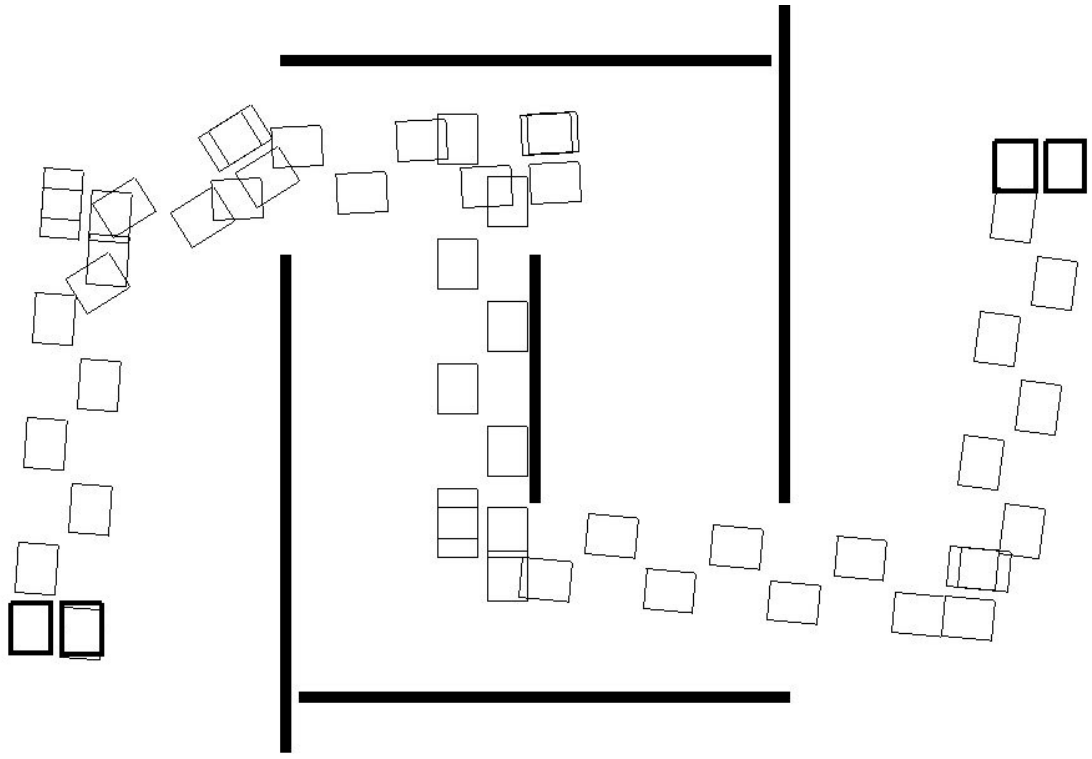

Рис. 15. Пять препятствий

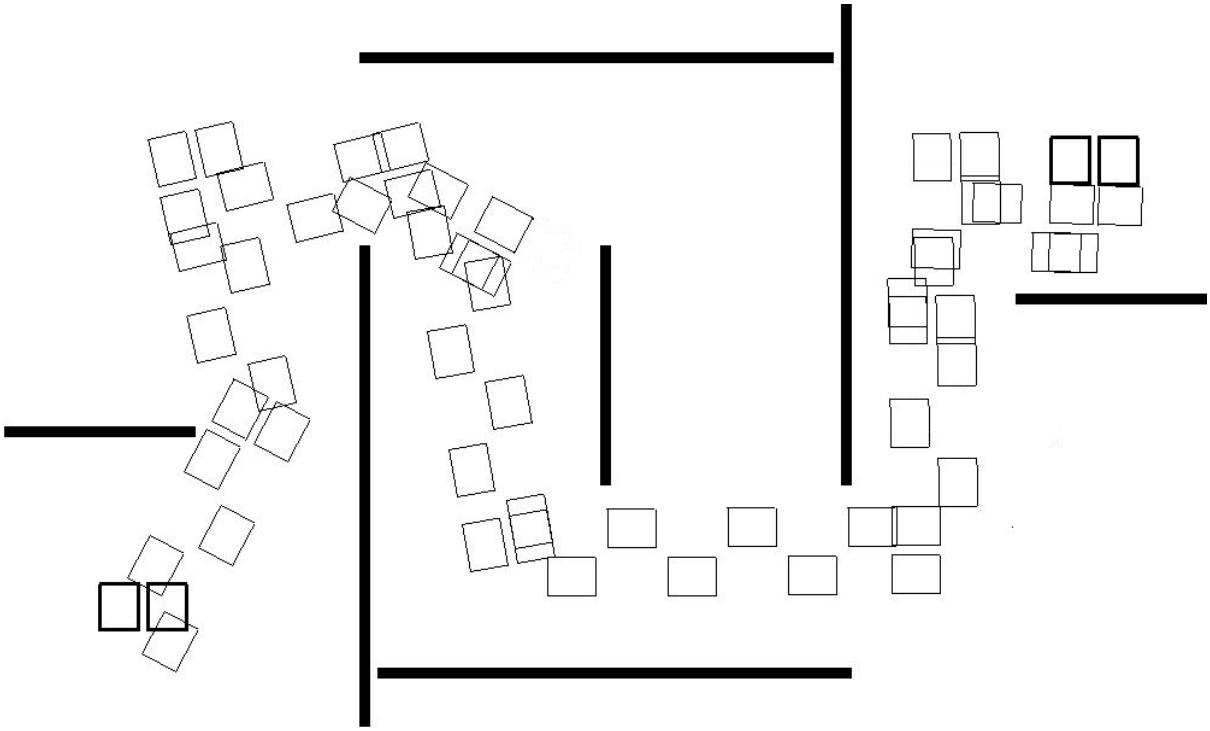

Рис. 16. Семь препятствий

В таблице 2 представлено процентное соотношение времени работы отдельных частей программы на каждой итерации алгоритма RRT.

Операция	Время выполнения
Генерация новой конфигурации	2%
Проверка на допустимость на данной карте	9%
Поиск ближайшей вершины дерева	16%
Проверка на достижимость	73%

Таблица 2. Процентное соотношение элементов расчетной программы

Заключение

В данной работе была рассмотрена задача построения траекторий антропоморфного механизма для его перемещения по заранее известной карте в аксиальной плоскости. Для решения этой задачи адаптирован и реализован вероятностный алгоритм RRT (rapidly exploring random trees) с оптимизацией посредством k-мерного дерева. Самой затратной частью работы программы стала проверка на существование пути между конфигурациями. Реализация и практическая применимость продемонстрированы в главах 2-3.

Список литературы

[1] Формальский А. М. Перемещение антропоморфных механизмов. М.: Наука, 1982. 368 с.

[2] Й. Виттенбург Динамика систем твердых тел. М.: Мир, 1980. 294 с.

[3] Белецкий В. В. Двуногая ходьба: модельные задачи динамики и управ- ления. М.: Наука, 1984. 288 с.

[4] LaValle S. M. Planning Algorithms. Cambridge University Press, 2006, 512 c.

[5] B. Tovar, A. Yershova, J. M. O'Kane, S. M. LaValle Information spaces for mobile robots // RoMoCo 2005, 2005.

[6] A. Yershova, S. M. LaValle. Improving motion planning algorithms by efficient nearest-neighbor searching // IEEE Transactions on Robotics, 23(1):151--157, February 2007.

[7] M. Arnold, Y. Baryshnikov, S. M. LaValle. Convex hull asymptotic shape evolution // In Proc. Workshop on the Algorithmic Foundations of Robotics, 2012.

[8] LaValle S. M. Rapidly-exploring random trees: A new tool for path planning // Technical Report (Computer Science Department, Iowa State University), 1998.

[9] Bourgeot J. M., Cislo N., Espiau B. Path-planning and tracking in a 3D complex environment for an anthropomorphic biped robot //Intelligent Robots and Systems, DOI: 10.1109/IRDS.2002.1041646.

[10] LaValle S. M., Kuffner J. J. RRT-Connect: An Efficient Approach to Single-Query Path Planning // In Proc. IEEE International Conference on Robotics and Automation, pp 995--1001, 2000.

[11] LaValle S. M., Kuffner J. J. Randomized kinodynamic planning // International Journal of Robotics Research. 2001. Vol. 20, No 5. pp. 378–400.

[12] Mark de Berg, Otfried Cheong, Marc van Kreveld , Mark Overmars Computational Geometry: Algorithms and Applications . Springer, 2008, 388 c.

[13] Kuffner J. J., Nishiwaki K., Kagami S., Inaba M., Inoue H. Footstep

planning among obstacles for biped robots // Intelligent Robots and Systems, DOI: 10.1109/IROS.2001.973406.

[14] Кормен Т. Х, Лейзерсон Ч. И, Ривест Р. Л, Штайн К. Алгоритмы: по- строение и анализ, 3-е издание. М.: «Вильямс», 2013, 1328 с.

[15] Armin Bruderlin, Thomas W. Calvert Goal-Directed, Dynamic Animation of Human Walking // Computer Graphics. 1989. №23.