УДК 004.896
ББК 30.1

РЕКОНФИГУРАЦИЯ ПРОСТРАНСТВЕННОГО ПОЛОЖЕНИЯ РОЯ РОБОТОВ

Ватаманюк И. В.[1], Панина Г. Ю.[2], Ронжин А. Л.[3]
(*ФГБУН Санкт-петербургский институт информатики и автоматизации РАН*)

Предложен алгоритм управления и навигации роя автономных гомогенных мобильных роботов, задействованных в формировании заданной пространственной выпуклой поверхности. Расчет траекторий роботов при реконфигурации роя проводится с учетом минимизации временных и энергетических затрат (*времени вычислений и суммарной длины траекторий*)*, а также контроля коллизий. Разработанный алгоритм имеет квадратичную вычислительную сложность. В ходе экспериментов по моделированию реконфигурации роя от 10 до 10000 роботов была проведена оценка числа коллизий, возникающих в случае прямолинейного движения роботов к своим таргетным точкам, при предположении, что роботы стартуют одновременно.*

Ключевые слова: рой роботов, топологическая робототехника, имитация поверхностей, групповое управление, беспилотные аппараты, робототехнические комплексы.

---

[1] *Ирина Валерьевна Ватаманюк, аспирант СПИИРАН* (*тел.:* (*812*) *328-70-81, vatamaniuk@iias.spb.su*).
[2] *Гаянэ Юрьевна Панина, доктор физико-математических наук, профессор, ведущий научный сотрудник лаборатории автономных робототехнических систем СПИИРАН* (*тел.:* (*812*) *328-70-81, gaiane-panina@rambler.ru*).
[3] *Андрей Леонидович Ронжин, доктор технических наук, профессор, заведующий лабораторией автономных робототехнических систем СПИИРАН* (*тел.:* (*812*) *328-70-81, ronzhin@iias.spb.su*).

1. Введение

Системы автономных мобильных роботов способны решать большой класс гражданских и военных задач, включая тушение пожаров, спасательные операции, аграрные и другие задачи в различных средах [1, 10, 13, 16]. Поэтому задачи реконфигурации роя роботов являются актуальными и достаточно широко изучаются и внедряются на практике. Используемые подходы основаны на различных принципах и, соответственно, базируются на различном математическом аппарате. Мы отсылаем читателя к работе [6], где представлен подробный обзор базовых алгоритмов и используемого математического аппарата. Как правило, в существующих работах учитываются взаимодействие роботов с соседями (граф взаимодействия агентов), а расчетные траектории движений представляют собой численные решения соответствующих дифференциальных уравнений. В отличие от традиционных работ, в данной статье мы предлагаем иной подход, существенно упрощающий вычислительную сложность задачи, но наносящий ущерб робастности алгоритма. Возможность его применения обусловлена спецификой конкретной задачи. Прежде чем описать его, рассмотрим существующие подходы и принципы решения схожих задач.

Применение многоагентных децентрализованных способов взаимодействия групп роботов позволяет создавать робототехнические системы, которые являются гибкими к реконфигурации, устойчивыми к противодействующим факторам и обладающими низкой себестоимостью [2, 7, 8, 9, 11, 12, 18].

В работе [11] изучается проблема разделения роя мобильных роботов на сбалансированные подгруппы, предлагаются алгоритмы контроля положения и ориентации групп роботов при выстраивании определенной пространственной структуры.

Работа [14] близка по постановке задачи к данной статье: совокупность роботов должна образовать некоторую пространственную фигуру. Авторы предлагают декомпозировать имитируемую фигуру на части и разделить рой на несколько подмножеств (групп) роботов, отвечающих за формирование своей части фигуры. После этого шага проблема управления роботами решается на двух уровнях: 1) взаимодействие роботов внутри

группы; 2) взаимодействие между группами роботов. При этом авторы используют стратегию «ведущий (лидер) – ведомый» и, в зависимости от сложности формируемой фигуры, вводят несколько уровней доминирования для ведомых групп роботов.

В работе [15] используется некоторый естественный аналог градиентного спуска. Каждый робот «видит» своих ближайших соседей и при реконфигурации движется в направлении, оптимизирующем требуемые целевые расстояния между роботами.

В работе [18] предложен новый топологический метод формирования пространственной фигуры из множества роботов на основе многоуровневой архитектуры, причем построение производится послойно с определенным расположением роботов на каждом уровне внутри всей формы.

Имеется ряд работ по самоорганизующимся многоагентым системам [1, 2]. Такие системы решают самые разнообразные задачи и (теоретически) могут выполнить реконфигурацию группы роботов, рассматриваемую в нашей статье. Однако специфика данной задачи позволяет произвести реконфигурацию пространственного расположения роя роботов с меньшими временными затратами на основе предложенного подхода.

Ниже (во втором разделе) представлена формальная постановка задачи реконфигурации роя роботов и перечислены основные этапы предложенного алгоритма. В разделе 3 приводится математическое описание алгоритма расчета траекторий движения роботов при реконфигурации. В разделе 4 приводится описание вспомогательных функций и процедур, используемых при работе алгоритма. Раздел 5 посвящен анализу результатов экспериментальной проверки алгоритма при моделировании реконфигурации роя от 10 до 10000 роботов.

2. Постановка задачи

В данной работе в результате реконфигурации роботам необходимо покрыть с заданной плотностью некоторую выпуклую поверхность. Выпуклость поверхности, а также отсутствие препятствий, придает задаче определенную специфику, в частности, позволяет предложить относительно простой алгоритм и сэкономить вычислительные и технические ресурсы.

Опишем подробнее постановку задачи. Мы имеем дело со следующими входными данными:

1. Имеется n идентичных объектов-роботов (*рой роботов*). Каждый из них представляет собой шар. Заданы координаты их центров.
2. R – радиус каждого из роботов.
3. *MinDist* – минимальное допустимое расстояние между центрами роботов.
4. S – некоторая выпуклая поверхность. Мы предполагаем, что поверхность задана двумя наборами данных:
   а) набором вершин $v_i = (x_i, y_i, z_i)$;
   б) набором треугольных граней, т.е. троек различных индексов (i, j, k), задающих грани поверхности. Мы считаем, что грани занумерованы от 1 до M. (Таким образом, M – число граней поверхности);
   в) данные пунктов а) и б) позволяют составить список ребер поверхности, т.е. набор всех ребер треугольных граней без повторений.
5. V – скорость движения роботов. Предполагается, что роботы могут либо двигаться со скоростью V, либо стоять неподвижно.
6. ρ – желаемая плотность размещения роботов на поверхности. А именно, ρ есть желаемое число объектов на квадрате 10×10.

В этих условиях требуется, избегая коллизий, переместить часть роботов так, чтобы они покрыли поверхность S с плотностью ρ. Точнее говоря, требуется получить следующие выходные параметры:

1) конечные координаты центров всех объектов, участвующих в реконфигурации;

2) расчетные траектории движения объектов, задействованных в моделировании фигуры;

3) оценку вычислительных затрат;

4) а также оценку времени перестроения объектов в требуемую фигуру.

При разработке алгоритма учитывались следующие технические условия:

1. Число объектов-роботов достаточно для покрытия поверхности с требуемой плотностью.

2. Плотность и минимальное допустимое расстояние связаны соотношением

(1) $\frac{\sqrt{\rho}}{MinDist} < \frac{1}{2}$.

Неформально говоря, это означает, что требуемая плотность не слишком велика.

3. Длина каждого из ребер триангуляции больше 2*MinDist*.

4. Угол между соседними гранями поверхности не меньше прямого.

В настоящей статье предлагается алгоритм реконфигурации роя, вычислительная часть которого имеет квадратичную сложность. Время реконфигурации существенно зависит от взаимного расположения роботов и поверхности. Нам представляется, что оно близко к минимально возможному, так как мы выбираем ближайших к цели роботов и предписываем траектории, близкие к прямолинейным. Чем меньше соотношение «радиус робота / перемещение робота», тем ближе траектории к прямолинейным, а значит, объективно оптимальным. Новизна алгоритма состоит в ряде приемов: назначении точек портала и пересечении поверхности траекториями по нормали, что позволяет получить почти прямолинейные траектории.

Теперь рассмотрим детально эти и другие особенности предлагаемого алгоритма реконфигурации роя роботов. Работа алгоритма начинается с проверки корректности входных данных и задания на формируемой поверхности сети точек с требуемой плотностью. Каждой точке сети приписывается некоторый объект-робот, и эта точка сети является *таргетной* (т.е. целевой) для данного робота. На рис. 1 показаны примеры траекторий движения роботов. Во избежание коллизий необходимо ввести следующие корректировки.

– Если роботу (на пути к таргетной точке) приходится пересекать поверхность, то он делает это в *точке портала*. Точки портала образуют некоторую сеть точек на поверхности, задаваемую алгоритмом, и служат для бесконфликтного перехода роботов из внутренней части пространства, ограни-

ченного поверхностью, во внешнюю. Рис. 1 показывает набор таргетных (черных) точек и набор (белых) точек портала.

– Роботы приближаются к поверхности по вектору нормали (см. рис. 1). Этот (новый) прием позволит избежать коллизий с уже прибывшими на свое место роботами. Таким образом, движение каждого робота состоит из (самое большее) четырех прямолинейных участков.

– Однако такое движение все еще может привести к коллизиям – ситуациям, когда расстояние между центрами роботов окажется меньше минимально допустимого *MinDist*. Поэтому некоторым роботам предпишем задержку, т.е. они начинают движение не в момент 0, а в момент *T*, свой для каждого робота.

На рис. 1 и 2 мы изображаем «плоскую» модель нашей задачи: роботы расположены на плоскости и после реконфигурации должны сформировать плоскую выпуклую фигуру, в данном случае – пятиугольник. Далее рассмотрим более детально предлагаемый алгоритм реконфигурации роя роботов.

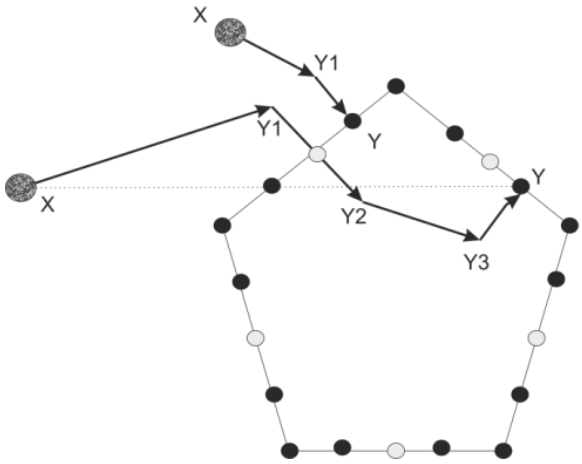

Рис. 1. Пример пересекающей и непересекающей поверхность траекторий. Черным цветом отмечены таргетные точки, белым - точки портала

3. Алгоритм расчета траекторий движения роботов при реконфигурации роя

Разработанный алгоритм состоит из следующих основных этапов: 1) анализ и оснащение формируемой поверхности и расчет координат таргетных точек и точек портала; 2) задание соответствия между роботами и таргетными точками; 3) расчет траекторий движения роботов с учетом возможных коллизий и времени начальных задержек каждого робота. Рассмотрим последовательно каждый из указанных этапов.

3.1. АНАЛИЗ И ОСНАЩЕНИЕ ФОРМИРУЕМОЙ ПОВЕРХНОСТИ

На первом этапе алгоритма выполняется расчет координат таргетных точек и точек портала. *Таргетные точки* – это те точки поверхности, в которые роботы должны прибыть в результате реконфигурации. В общем случае таргетные точки, а также их число, зависят от самой поверхности и от заданной плотности. Назначение таргетных точек на поверхности производится следующим образом:

1. Вычислим желаемое расстояние между таргетными точками на поверхности:

$$\delta = \frac{10}{\sqrt{\rho}}. \quad (2)$$

Таргетные точки условно делятся на три серии.

2. Первая серия таргетных точек – вершины поверхности.

3. Вторая серия таргетных точек – это точки, лежащие на ребрах; они задаются следующим способом:

Переберем все ребра по одному. Для каждого ребра зададим несколько таргетных точек; они расположены равномерно вдоль ребра. А именно:

а) пусть вершины ребра – точки $A = (a_1, a_2, a_3)$, $B = (b_1, b_2, b_3)$.

Вычислим $N = [\mathrm{dist}(A, B) / \delta]$. Здесь и далее квадратные скобки означают взятие целой части;

б) **for** $i = 1$ **to** $N - 1$ **do** Задаем таргетную точку $T = A + i\frac{(B-A)}{N}$.

4. Прежде чем задать третью серию таргетных точек, зададим точки портала. На каждой грани поверхности будет по одной точке портала. В дальнейшем число точек портала можно увеличить. Но пока придерживаемся следующих действий:

Переберем все грани по одной.

Путь вершины грани F – точки: $A = (a_1, a_2, a_3)$, $B = (b_1, b_2, b_3)$, $C = (c_1, c_2, c_3)$.

а) зададим точку портала грани F:

(3) $Portal = \frac{1}{3}(A + B + C) = (\frac{1}{3}(a_1,b_1,c_1), \frac{1}{3}(a_2,b_2,c_2), \frac{1}{3}(a_3,b_3,c_3))$;

б) зададим таргетные точки третьего типа, лежащие строго внутри этой грани:

i. вычислим

(4) $N_1 = \left[\frac{\text{dist}(A, B)}{\delta}\right]$,

(5) $N_2 = \left[\frac{\text{dist}(A, C)}{\delta}\right]$;

ii. **for** $i = 1$ **to** $N_1 - 1$ **do**

for $j = 1$ **to** $N_2 - i\left[\frac{N_2}{N_1}\right] - 1$ **do**

Зададим таргетную точку $T = A + i\frac{(B-A)}{N_1} + j\frac{(C-A)}{N_2}$.

Возможно, некоторые из таргетных точек расположены недопустимо близко друг к другу или к точке портала. Для их устранения используем следующий порядок действий:

1. Переберем все вершины поверхности и удалим те из таргетных точек, которые оказались ближе к данной вершине, чем минимальное допустимое расстояние.

2. Переберем все точки портала и удалим те из (оставшихся) таргетных точек, которые оказались ближе к данной точке портала, чем минимальное допустимое расстояние.

3. Переберем все оставшиеся таргетные точки второго и третьего типа и удалим те из (оставшихся) таргетных точек, которые оказались ближе к данной точке, чем минимальное допустимое расстояние.

После того как точки портала и таргетные точки определены, производится оснащение поверхности системой нормалей и

нормальными точками. Для каждой из таргетных точек и точек портала зададим по две *нормальные точки* (см. рис. 2). Каждая нормальная точка отстоит от исходной в направлении нормали на расстояние *MinDist*.

При оснащении точек портала и таргетных точек третьего типа перебираются по очереди все грани поверхности. Пусть вершина грани F – точки $A = (a_1, a_2, a_3)$, $B = (b_1, b_2, b_3)$, $C = (c_1, c_2, c_3)$, далее:

1. Зададим вектор нормали к грани:

(6) $\vec{n}(F)=\frac{\overrightarrow{AB}\times\overrightarrow{AC}}{|\overrightarrow{AB}\times\overrightarrow{AC}|}$,

где × – векторное произведение.

2. Добьемся нужной ориентации (вектор нормали должен смотреть «наружу»):

а) зададим две точки:

(7) $N(Portal) = Portal + \infty\cdot\vec{n}(F)$,

(при программной реализации знак бесконечности заменяется константой достаточно большой величины);

(8) $Norm(Portal) = Portal + MinDist\cdot\vec{n}(F)$,

б) запускаем

(9) $IntersectS(Norm(Portal), N(Portal))$;

в) **if** $Count \neq 0$ **and** $Count \neq 2$

begin

(10) $\vec{n}(F) = -\vec{n}(F)$

end.

3. Зададим нормальные точки для точки портала этой грани правилом

(11) $Norm_1(Portal) = Portal + MinDist\cdot\vec{n}(F)$,

(12) $Norm_2(Portal) = Portal - MinDist\cdot\vec{n}(F)$.

4. Переберем по очереди все точки третьего типа, принадлежащие данной грани. Для каждой точки T зададим нормальные точки правилом

(13) $Norm_1(T) = T + MinDist\cdot\vec{n}(F)$,

(14) $Norm_2(T) = T - MinDist\cdot\vec{n}(F)$.

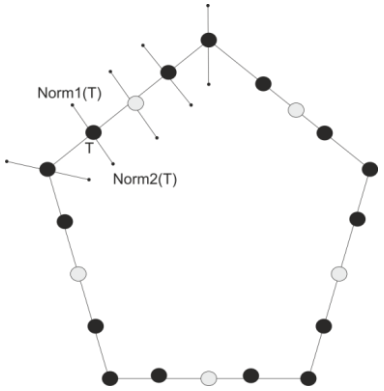

Рис. 2. Задание на поверхности системы нормалей и нормальных точек.

При оснащении таргетных точек второго типа перебираются по очереди все ребра поверхности.

Для каждого ребра E с вершинами AB

1. Находим ABC и ABD – примыкающие к ребру грани.
2. Зададим вектор

(15) $\vec{n}(E)=\frac{\vec{n}(ABC)+\vec{n}(ABD)}{|\vec{n}(ABC)+\vec{n}(ABD)|}$.

3. Переберем по очереди все таргетные точка второго типа, принадлежащие ребру AB. Для каждой точки T зададим нормальные точки правилом

(16) $Norm_1(T) = T + MinDist\cdot\vec{n}$,

(17) $Norm_2(T) = T - MinDist\cdot\vec{n}$.

При оснащении таргетных точек первого типа перебираются по очереди все вершины (они же точки первого типа). Для каждой из них:

1. Пусть таргетная точка T – некоторая вершина. Выберем три различные грани F_1, F_2, F_3, примыкающие к этой вершине.
2. Зададим вектор

(18) $\vec{n}(T)=\frac{\vec{n}(F_1)+\vec{n}(F_2)+\vec{n}(F_3)}{|\vec{n}(F_1)+\vec{n}(F_2)+\vec{n}(F_3)|}$

3. Зададим нормальные точки правилом

(19) $Norm_1(T) = T + MinDist\cdot\vec{n}$,

(20) $Norm_2(T) = T + MinDist\cdot\vec{n}$.

3.2. ЗАДАНИЕ СООТВЕТСТВИЯ МЕЖДУ РОБОТАМИ И ТАРГЕТНЫМИ ТОЧКАМИ

Следующим этапом алгоритма является присвоение каждому роботу, задействованному в реконфигурации, своей таргетной точки следующим образом:

1. Упорядочим линейно таргетные точки (т.е. занумеруем их числами от 1 до N).

2. Определим, какой именно объект-робот направится к каждой из таргетных точек. Для этого переберем таргетные точки по порядку номеров, и каждой из них припишем ближайший объект-робот. Точнее говоря, проделаем следующее:
   а) для таргетной точки номер 1 находим ближайший робот, присваиваем ему номер 1, занося робот в отдельный список *действующих роботов* под номером 1;
   б) для таргетной точки номер 2 находим ближайший *из оставшихся роботов*, присваиваем ему номер 2, занося его в список действующих роботов под номером 2;
   в) и так далее для всех таргетных точек.

3. Таким образом, у нас есть упорядоченный список действующих роботов. Именно они будут участвовать в реконфигурации роя. Возможно, остались «лишние» роботы, в реконфигурации не участвующие. Мы будем называть их *пассивными роботами*. У каждого действующего робота тем самым имеется:
   а) порядковый номер k;
   б) тройка координат, задающих начальную позицию его центра;
   в) тройка координат, задающих его таргетную точку. В эту точку должен быть перемещен центр робота в результате реконфигурации.

4. Для каждого робота в дальнейшем определим:
   а) число *NumberPathSegments* прямолинейных участков пути, которое предстоит выполнить роботу. Оно равно двум или четырем (см. рис. 1);
   б) траекторию робота. Она задается перечислением промежуточных точек;
   в) время задержки отправления робота.

3.3. РАСЧЕТ ТРАЕКТОРИЙ ДВИЖЕНИЯ РОБОТОВ

Перед расчетом траекторий выполняется сортировка действующих роботов на два типа. К первому типу относятся те, чья траектория не пересекает поверхность; ко второму типу – остальные роботы. Далее реконфигурация роя производится в следующем порядке.

Действующий робот первого типа с номером k в момент времени $T(k)$ начинает двигаться к нормальной точке своей таргетной точки. Добравшись до нее, он (без задержки) двигается к своей таргетной точке, в которой и останавливается.

Действующий робот второго типа с номером k в момент времени $T(k)$ начинает двигаться к нормальной точке портала, пересекает поверхность по нормали, затем начинает двигаться к нормальной точке своей таргетной точки. Добравшись до нее, он (без задержки) двигается к своей таргетной точке, в которой и останавливается.

Зададим траектории каждого робота следующим образом: Пусть X – стартовая точка робота, Y – таргетная точка.

Запускаем процедуру *IntersectS*(X, $Norm_1(Y)$) (см. раздел 4)

if *Count* = 0 **then**

begin

$Y_1 = Norm_1(Y)$.

Назначаем роботу X траекторию: $X \rightarrow Y_1 \rightarrow Y$.

Присваиваем значение *NumberPathSegments* := 2.

Завершаем назначение траектории для этого робота.

end

else

Запускаем процедуру *IntersectS*(X, $Norm_2(Y)$)

if *Count* ≠ 0 **then**

begin

$Y_1 = Norm_2(Y)$

Назначаем роботу X траекторию: $X \rightarrow Y_1 \rightarrow Y$.

Присваиваем значение *NumberPathSegments* := 2.

Завершаем назначение траектории для этого робота.

end

else

$I \coloneqq FaceNumber$

if $Count = 1$ **then**

begin

Запускаем $IntersectS(X, Norm_1(Portal(I)))$

if $Count = 0$ **then**

begin

$Y_1 = \left(Portal(I)\right)$.

$Y_2 = Norm_2(Portal(I))$.

$Y_3 = Norm_2(Y)$.

Назначаем роботу X траекторию: $X \to Y_1 \to Y_2 \to Y_3 \to Y$.

Присваиваем значение $NumberPathSegments(X) = 4$.

Завершаем назначение траектории для этого робота.

end

else

begin

$Y_1 = \left(Portal(I)\right)$.

$Y_2 = Norm_1(Portal(I))$.

$Y_3 = Norm_2(Y)$.

Назначаем роботу X траекторию: $X \to Y_1 \to Y_2 \to Y_3 \to Y$.

Присваиваем значение $NumberPathSegments(X) = 4$.

Завершаем назначение траектории для этого робота.

end

end.

Теперь опишем, как рассчитывается момент времени $T(k)$ начала движения каждого робота. Введем вспомогательную величину – *квант времени*:

(21) $\tau = \frac{2MinDist}{V}$.

Это время, за которое робот проходит расстояние $2MinDist$. Вначале все задержки начального движения роботов полагаются равными нулю.

For $k = 2$ **to** N

begin

Тестируем робот с номером k на наличие коллизий с роботами с меньшими номерами. В случае коллизий робот со старшим номером уступает дорогу. А именно:

for $i = 1$ **to** $k - 1$ **do**
begin
if *CollisionTest*(i, k) = TRUE **then**
1. $T(k) = T(k) + \tau$.
2. Запускаем заново тест для k-го робота для младших номеров начиная с $i = 1$.

end
end.

Теперь оценим сложность предложенного алгоритма, исходя из естественного предположения, что общее число роботов (и активных, и пассивных) соразмерно с числом таргетных точек (или, что то же самое, с числом активных роботов). Очевидно, сложность каждой из ступеней алгоритма не более чем квадратична (относительно числа таргетных точек). Следовательно, мы имеем квадратичный алгоритм, сложность которого не зависит ни от времени потенциального перемещения роботов, ни от расстояний. Деликатным моментом в оценке сложности алгоритма оказывается разрешение коллизий, создающее циклы. Здесь мы пользуемся следующим эвристическим соображением в сочетании с результатами экспериментов (см. раздел 5): поскольку число конфликтующих пар роботов не превышает 2,22% от общего числа пар активных роботов, общее число циклов разрешения коллизий растет не быстрее, чем логарифм от общего числа пар роботов и, следовательно, вносит пренебрежимо малый вклад в общую сложность.

4. Вспомогательные процедуры проверки возможных коллизий роботов

Здесь опишем ряд вспомогательных функций и процедур, использованных в третьем разделе. Первые две из них определяют наличие пересечения отрезка и поверхности; вторые две – тестируют движения роботов на наличие коллизий.

Булева функция *IntersectTriangle* определяет наличие пересечения отрезка и треугольника, лежащих в трехмерном пространстве. На вход подаются отрезок *DE* и треугольник *ABC*.

IntersectTriangle(*D*, *E*, *A*, *B*, *C*) определяет, пересекаются ли отрезок и треугольник:

begin

IntersectTriangle = FALSE.

begin

if

(22) *Sign*(*DET*(*D–A*, *B–A*, *E–A*)) = *Sign*(*DET*(*D–B*, *C–B*, *E–B*))= *Sign*(*DET*(*D–C*, *B–C*, *E–C*))

and

(23) *Sign*(*DET*(*B–A*,*C–A*,*D–A*)) = –*Sign*(*DET*(B–A,C–A,E–A))

then

IntersectTriangle = TRUE.

end

end.

Процедура *IntersectS* определяет наличие пересечений отрезка с поверхностью *S*. На вход подается отрезок *DE*. Считается, что при этом поверхность *S* задана априори.

Процедура выдает два целых числа:

1. Число *Count*(*D, E*) говорит, сколько раз отрезок *DE* пересекает поверхность *S*. Оно принимает значения 0, 1, или 2 (в силу выпуклости поверхности).
2. Если пересечение не пусто, то *FaceNumber* - номер грани, содержащей точку пересечения. (Примечание: такая точка может быть не одна. В силу особенности алгоритма нам подойдет любая).

В процессе работы процедура перебирает все грани поверхности и запускает для каждой грани *IntersectTriangle*.

Функция *SmallCollisionTest*. Пусть имеются два робота. Известны положения каждого из них в момент времени t_0. Каждый из роботов движется от момента времени t_0 до момента времени t_1 с постоянной скоростью (она равна либо нулю, либо *V*). Известны координаты точки, в направлении которой движется каждый из роботов. Функция *SmallCollisionTest* определяет наличие коллизии на временном промежутке (t_0, t_1).

Работа функции сводится к поиску минимума квадратной функции от времени (расстояния между роботами) на некотором временном интервале, поэтому конкретные расчеты нами опущены в силу их элементарности.

Процедура *CollisionTest* тестирует траектории движения двух отдельно взятых роботов и работает следующим образом. Имеются два робота. Каждый из них начинает движение в (свой) момент времени по предписанной траектории, состоящей из (самое большее) четырех прямолинейных участков. Булева функция *CollisionTest* определяет наличие коллизии между этими двумя роботами следующим образом:

1. Для обоих роботов рассчитываются временные интервалы, соответствующие прямолинейным участкам траектории.
2. Для всех получившихся пересечений временных интервалов применяется функция *SmallCollisionTest*.
3. Если хотя бы для одного временного интервала определяется наличие коллизии, т.е. функция *SmallCollisionTest* возвращает *TRUE*, то процедура *CollisionTest* возвращает *TRUE*.

5. Компьютерное моделирование: результаты экспериментов

Основная цель проведенных экспериментов состояла в оценке числа коллизий, возникающих в том случае, когда роботы движутся к своим таргетным точкам прямолинейно, начиная движение одновременно. При помощи данной оценки были сделаны выводы о сложности предложенного алгоритма. Для проведения экспериментов была смоделирована поверхность в форме граней куба. Эксперименты проводились для следующих значений числа роботов: 10, 50, 100, 500, 1000, 5000, 10000. Были рассмотрены три варианта размещения роботов: при максимально возможной плотности размещения ρ_{max}, при плотности, равной $0{,}5\rho_{max}$, а также при плотности $0{,}25\rho_{max}$. Размер ребра куба в каждом случае был рассчитан исходя из плотности размещения роботов на поверхности. Для каждого случая рассматривались два варианта

начального расположения роботов: вокруг центра куба на расстоянии не большем, чем 10 длин ребра куба и вне поверхности на расстоянии, не меньшем 10 длин ребра куба. Были оценены: время расчета траекторий (рис. 3(а)), количество коллизий при максимальной плотности ρ_{max} (рис. 3(б)), а также количество коллизий при различных значениях плотности размещения роботов (рис. 4). Кроме того, на рис. 4 для наглядности представлен график количества всех возможных сочетаний пар роботов.

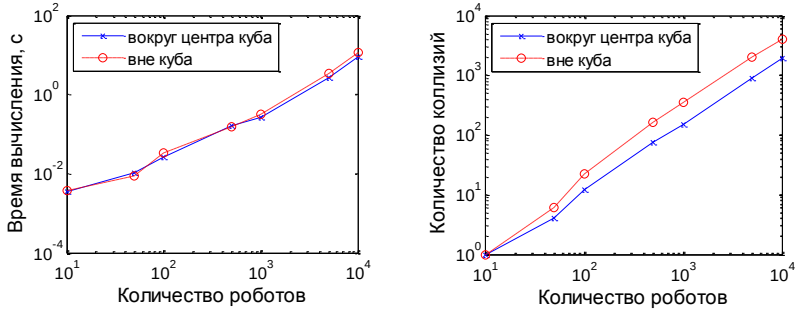

Рис. 3. (*а*) *оценка времени расчета всех траекторий;*
(*б*) *оценка количества коллизий при* ρ_{max}

Как видно из рис. 3(б), число коллизий при указанных условиях не превышает 2,22% от всех возможных пар роботов. На рис. 3(а) представлены средние значения времени вычисления траекторий. Заметим, что время расчета траекторий растет пропорционально количеству роботов и практически не зависит от начального расположения роботов.

Из рис. 4 видно, что с уменьшением плотности расположения роботов количество коллизий уменьшается пропорционально, вне зависимости от способа начального расположения роботов. Причем число коллизий во всех экспериментах не превышает 2,22% от всех возможных пар роботов.

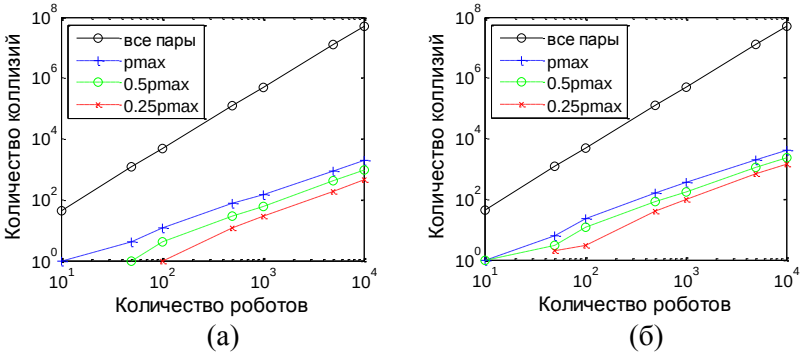

Рис. 4. Оценка количества коллизий при расположении роботов: (*а*) *вокруг центра куба;* (*б*) *вне куба.*

6. Заключение

В статье предложен алгоритм, обладающий следующими особенностями: 1) траектории движения почти прямолинейны, следовательно, энергозатраты на перемещения роботов практически минимальны; 2) естественно ожидать, что с вычислительной точки зрения алгоритм будет работать быстрее, чем метод градиентного спуска, так как отсутствует многоступенчатый расчет траекторий, неизбежный в последнем (отметим, однако, что экспериментально эти ожидания нами не проверялись); 3) направление движения каждого из роботов меняется самое большее трижды, причем незначительно, что экономит затраты на пространственную переориентацию робота.

Разработанная программная модель позволяет наглядно продемонстрировать движение большого числа малых робототехнических комплексов при формировании пространственных фигур. При моделировании движения роя разработанный алгоритм экспериментально проверен при разном количестве роботов (от 10 до 10000 единиц) с целью оценивания масштабируемости математического обеспечения и объема потребляемых вычислительных ресурсов. Предлагаемые решения управления движением роя роботов могут быть применены в быстро развер-

тываемых имитационных средствах, а также распределенных киберфизических системах [3, 4, 5, 17].

Одной из наиболее актуальных практических задач, где будут использованы разработанные алгоритмы, является разработка методики конфигурации автоматизированных робототехнических средств, позволяющих осуществить транспортировку пострадавшего в определенной позе, оптимальной для его состояния. Другой практической задачей является применение роя гомогенных роботов для формирования поверхности имитируемой пространственной фигуры, позволяющей средствам удаленной регистрации обнаружить заданные классы объектов.

Работа выполнена в рамках проекта программы Президиума РАН I.40П «Актуальные проблемы робототехники».

Литература

1. ГОРОДЕЦКИЙ В.И. *Самоорганизация и многоагентные системы. II. Приложения и технология разработки* // Известия РАН. Теория и системы управления. – 2012. – №3. – С. 55–75.
2. ИВАНОВ Д.Я. *Формирование строя группой беспилотных летательных аппаратов при решении задач мониторинга* // Известия Южного федерального университета. Технические науки. – 2012. – Вып.4, том 129. – С. 219-224.
3. КРЮЧКОВ Б.И., КАРПОВ А.А., УСОВ В.М. *Перспективные подходы к применению сервисных роботов в области пилотируемой космонавтики* // Труды СПИИРАН. – 2014. – Вып.32. – С. 125–151.
4. МИРОНОВ В.И., МИРОНОВ Ю.В., ПРИЩЕПА Ю.В. *О некоторых способах улучшения точностных характеристик координатной привязки изображений, получаемых беспилотными летательными аппаратами* // Труды СПИИРАН. – 2009. – Вып.9. – С. 159–167.
5. РОНЖИН А.Л., ЮСУПОВ Р.М. *Многомодальные интерфейсы автономных мобильных робототехнических комплексов* // Известия Южного федерального университета. Технические науки. – 2015. – №1(162). – С. 195–206.
6. BULLO F., CORTES J., MARTINEZ S. *Distributed Control of Robotic Networks*. – Princeton University Press, 2009. – 314 p.

7. CHEN Y.Q., WANG Z. *Formation control: a review and a new consideration* // Proc. IEEE Int. Conf. Intell. Robots Syst. – 2005. – P. 3181–3186.
8. EFRIMA A., PELEG D. *Distributed algorithms for partitioning a swarm of autonomous mobile robots* // Theoretical Computer Science. – 2009. – Vol.410. – P. 1355–1368.
9. GAZI V. *Swarms aggregation using artificial potentials and sliding mode control* // IEEE Transactions on Robotics. – 2005. – Vol.21(4). – P. 1208–1214.
10. HAGHIGHI R., CHEAH C. *Multi-group coordination control for robot swarms* // Automatica. – 2012. – Vol.48. – P. 2526–2534.
11. HSIEH M.A., KUMAR V., CHAIMOWICZ L. *Decentralized controllers for shape generation with robotic swarms* // Robotica. – 2008. – Vol.26(5). – P. 691–701.
12. HU J., XU J., XIE L. *Cooperative search and exploration in robotic networks* // Unmanned Systems. – 2013. – Vol.01. – P. 121–142.
13. LAWTON J.R., BEARD R.W., YOUNG B.J. *A decentralized approach to formation maneuvers* // IEEE Transactions on Robotics and Automation. – 2003. – Vol.19(6). – P. 933–941.
14. REN W., SORENSEN N. *Distributed coordination architecture for multirobot formation control* // Journal of Robotics and Autonomous Systems. – 2008. – Vol.56(4). – P. 324–333.
15. SHUANG L., DONG S., CHANGAN Z. *Coordinated motion planning for multiple mobile robots along designed paths with formation requirement* // IEEE/ASME Transactions on Mechatronics. – 2011. – Vol.16. – P. 1021–1031.
16. SOUSSELIER T., DREO J., SEVAUX M. *Line formation algorithm in a swarm of reactive robots constrained by underwater environment* // Expert Systems with Applications. – 2015. – [Электронный ресурс] – URL: http://dx.doi.org/10.1016/j.eswa.2015.02.040. – (дата обращения 20.11.2015.)
17. STANKEVICH L., SEREBRYAKOV S., IVANOV A. *Data mining techniques for RoboCup soccer agents* // Proc. AIS-ADM'2005. – Vol. 3505 LNAI, 2005. – P. 289–301.

18. YAN X., CHEN J., SUN1 D. *Multilevel-based topology design and shape control of robot swarm* // Automatica. – 2012. – Vol. 48. – P. 3122–3127.

RECONFIGURATION OF ROBOT SWARM FORMATION

Irina Vatamaniuk, St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences, St. Petersburg, PhD student (St. Petersburg, 14 Line St., 39, (812) 328-70-81, vatamaniuk@iias.spb.su).

Gayane Panina, St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences, St. Petersburg, Doctor of Science, professor, senior researcher (St. Petersburg, 14 Line St., 39, (812) 328-70-81, gaiane-panina@rambler.ru).

Andrey Ronzhin, St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences, St. Petersburg, Doctor of Science, professor, deputy director for research (St. Petersburg, 14 Line St., 39, (812) 328-70-81, ronzhin@iias.spb.su).

Abstract: *An algorithm of control and navigation for a swarm of autonomous homogeneous mobile robots involved in the formation of a convex space surface is discussed. Robot trajectories during reconfiguration minimize the cost of time and energy (computation time and the total length of robot paths), taking into account collision control. The developed algorithm has quadratic computational complexity. During the experiments reconfiguration of a swarm with various numbers of robots* (*from 10 to 10,000 robots*) *was modeled. The number of collisions was evaluated assuming massed start of all robots.*

Keywords: robot swarm, topological robotics, surface imitation, group control, unmanned vehicles, robotic systems.

Статья представлена к публикации членом редакционной коллегии Б.Т. Поляком

Поступила в редакцию 03.06.2015.
Опубликована 30.11.2015.