ISSN 1814-1196
Научный вестник НГТУ
том 55, № 2, 2014, с. 109–121

http://journals.nstu.ru/vestnik
Scientific Bulletin of NSTU
Vol. 55, No. 2, 2014, pp. 109–121

СОВРЕМЕННЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

MODERN INFORMATION TECHNOLOGIES

УДК 004.896:004.42

Приближённые алгоритмы локализации мобильного робота*

ДАО ЗУЙ НАМ[1], С.А. ИВАНОВСКИЙ[2]

[1] 197376, РФ, г. Санкт Петербург, ул. Профессор Попова, 5, Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина), аспирант, e-mail: duynam81@mail.ru
[2] 197376, РФ, г. Санкт Петербург, ул. Профессор Попова, 5, Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина), к. т. н., доцент, e-mail: saivanovsky@mail.ru

Рассматриваются два приближённых алгоритма локализации мобильного робота, снабженного картой в виде простого многоугольника без дыр. Гипотезам локализации соответствуют экземпляры карты с отметкой предполагаемого положения робота. Робот должен определить свое истинное начальное местоположение, перемещаясь и обозревая видимую окрестность, чтобы устранить все неправильные гипотезы. При этом суммарная длина перемещений робота должна быть минимальной. Оптимизационная задача локализации робота является *NP*-полной, поэтому рассматриваются приближенные алгоритмы. Один из алгоритмов основан на использовании триангуляции простого многоугольника, представляющего карту. Предобработка в виде триангуляции простого многоугольника позволяет эффективно реализовать основные действия алгоритма, такие как, например, вычисление многоугольника видимости, поиск кратчайшего пути в многоугольнике между двумя точками, отсечение лишних гипотез. Второй алгоритм использует оверлей (пересечение) экземпляров карты. В пересечении выделяются так называемые окна, «заглядывая» в которые робот отсекает ложные гипотезы. На основе программной реализации нескольких алгоритмов проведено их экспериментальные исследование, использующее сгенерированные модельные карты. Приведены численные результаты машинных экспериментов и дана их интерпретация. Предлагаемые алгоритмы по критерию минимизации длины пройденного роботом пути незначительно уступают известным ранее алгоритмам, но на модельных примерах работают быстрее. Анализируются возможные способы уменьшения времени работы алгоритмов за счет использования параллелизма.

Ключевые слова: вычислительная геометрия, робототехника, мобильный робот, локализация робота, простой многоугольник, многоугольник видимости, генерация гипотез, проверка гипотез, пересечение полигонов, триангуляция полигона, сложность алгоритма, приближённый алгоритм, генерация карты, экспериментальное исследование алгоритма

ВВЕДЕНИЕ

Развитие аппаратной базы разнообразных мобильных робототехнических устройств (в том числе их сенсорного оснащения) стимулировало растущий интерес к широкому кругу прикладных задач (от применения в промышленности и медицинских клиниках до микро- и нанороботов), к методам их решения, к их математическому и программному обеспечению. Многие из этих задач лежат на стыке робототехники с другими научно-техническими дисциплинами, например, с компьютерными технологиями [1] и алгоритмическими задачами вычислительной геометрии [2]. Одна из таких задач – это задача локализации мобильного робота, который перемещается во внешней среде и снабжен моделью (картой) этой среды [1].

* *Статья получена 24 декабря 2013 г.*

Задача локализации мобильного робота состоит в определении координат робота в системе отсчета, связанной с внешней средой. Робот снабжен картой внешней среды в виде плоского простого многоугольника P с n вершинами без отверстий. Мобильный робот помещен в заранее неизвестное место p в пределах P (рис. 1). Для решения задачи локализации робот, во-первых, должен, обозревая свою окрестность и соотнося полученный многоугольник видимости $V = V(p)$ с картой, определить, является ли его начальное местоположение единственным. Затем на основании анализа многоугольников P и V робот должен сгенерировать множество H всех гипотез о своем местоположении $p_i \in P$ таким образом, что область видимости в точке p_i конгруэнтна V. Далее робот должен определить свое истинное начальное местоположение, перемещаясь и обозревая окрестность, чтобы устранить все неправильные гипотезы о своем местоположения. При этом суммарная длина перемещений робота должна быть минимальной.

На рис. 1 приведены многоугольник карты P (слева), многоугольник видимости V (в центре) и два возможных начальных местоположения p_1 и p_2 (справа).

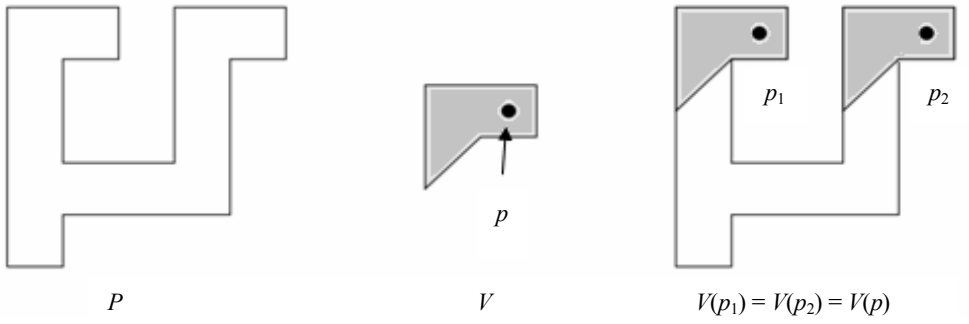

Рис. 1. Многоугольник карты P (слева), многоугольник видимости V (в центре) и два возможных начальных местоположения p_1 и p_2 (справа)

Оптимизационная задача локализации мобильного робота является *NP*-трудной задачей [1]. Поэтому в основном рассматриваются приближённые (полиномиальные по вычислительной сложности) алгоритмы локализации мобильного робота [3], [4], [5], [6]. При этом, как правило, внимание акцентируется на характеристиках, описывающих отклонения от стоимости оптимального решения (суммарной длины перемещений робота). Характеристики вычислительной сложности таких алгоритмов оцениваются асимптотически, а данные о реальном времени работы алгоритмов, как правило, не приводятся, в том числе по причине высокой вычислительной сложности алгоритмов, например, $O(n^5 \log n)$[4] или $O(n^{12})$ [6].

Далее будут рассмотрены приближенные алгоритмы, которые, предположительно, обеспечивают по сравнению с известными алгоритмами несколько меньшую, но сопоставимую точность полученного решения (длины пути перемещения робота), зато имеют меньшую вычислительную сложность и время реальной работы при характерных размерах решаемой оптимизационной задачи. Предлагаются два таких (в известном смысле упрощенных) алгоритма локализации: (1) алгоритм на основе триангуляции карты и (2) алгоритм с выделением окон в многоугольнике пересечения экземпляров карты, соответствующих разным гипотезам. Для сравнения вычислительной эффективности алгоритмов используется компьютерное экспериментальное исследование.

1. ОБЩАЯ СХЕМА РЕШЕНИЯ ЗАДАЧИ ЛОКАЛИЗАЦИИ

Известные алгоритмы локализации мобильного робота [3], [4], [5], [6] включают две фазы: генерацию гипотез и проверку гипотез. В фазе генерации гипотез вычисляется множество гипотетических местоположений робота $p_1, p_2, ..., p_k \in P$, которые соответствуют наблюдениям робота в его начальном положении. В фазе проверки гипотез исключаются неправильные гипотезы.

В фазе генерации гипотез строится множество гипотез $H = \{h_1, h_2, ..., h_k\}$, $(\forall i \in 1..k \mid h_i : p = p_i)$, определяемых гипотетическими местоположениями в P, в которых робот мог бы быть расположен первоначально. Далее выбирается произвольное гипотетическое p_i местоположение из H, чтобы служить точкой привязки. Затем для каждого гипотетического местоположения p_j, $1 \le j \le k, j \ne i$ определяется вектор переноса $t_j = p_i - p_j$, который соответствует переводу местоположения p_j в p_i. $(p_i = p_j + t_j)$. Вычисляются копии P_1, P_2,…, P_k многоугольника карты P, соответствующие множеству гипотез H таким образом, что P_j преобразуется в $P = P_i$ смещением на t_j (можно считать, что при этом экземпляр P_i преобразуется в себя смещением на $t_i = 0$). Точка гипотетического местоположения p_j в каждом многоугольнике P_j переходит при таком переносе в точку p_i многоугольника $P = P_i$.

Определим теперь *оверлей* (*наложение*) *многоугольников* (многоугольники рассматриваются здесь как подразбиение плоскости на внутреннюю и внешнюю грани, а стороны многоугольников рассматриваются как ребра такого подразбиения плоскости) [2], [7].

Определение. Оверлей многоугольников $P_1, P_2, ..., P_k$ относительно выделенного экземпляра многоугольника карты $P = P_i$ является структурой, полученной *объединением* множеств ребер всех смещенных многоугольников P_j, $1 \le j \le k$. Это множество ребер порождает разбиение плоскости на грани.

В более общем случае можно определить оверлей (наложение) *Overlay*(S_1,S_2) двух *подразбиений* S_1 и S_2, как подразбиение *Overlay*(S_1,S_2), такое, что существует грань F в *Overlay*(S_1,S_2), если и только если имеются грани F_1 в S_1 и F_2 в S_2, такие, что F является максимальным связным подмножеством в $F_1 \cap F_2$ [2]. На рис. 1 приведен пример оверлея многоугольников $Overlay(P_1, P_2)$.

Пересечение многоугольников $P_1, P_2, ..., P_k$ относительно выделенного экземпляра многоугольника карты $P = P_i$ определим как пересечение смещенных многоугольников P_j, $1 \le j \le k$. Это пересечение является гранью (или набором граней) оверлея многоугольников. На рис. 2 грань оверлея, которая является пересечением $\text{Intersection}(P_1, P_2)$ многоугольников, заштрихована.

Определение. *Внутреннее ребро* оверлея многоугольников является таким ребром их пересечения (одним из нескольких), которое отделяет область пересечения от других внутренних граней оверлея, в противоположность тем ребрам, которые принадлежат пересечению, но отделяют область пересечения от внешней грани оверлея на двумерной плоскости (см. рис. 3, на котором приведены внутренние ребра e_1 и e_2).

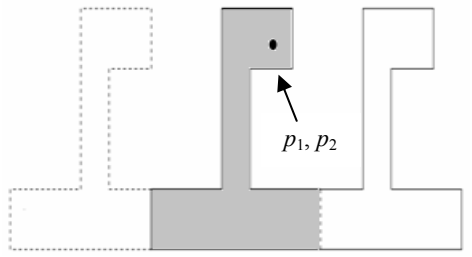

Рис. 2. Оверлей, который является пересечением многоугольников

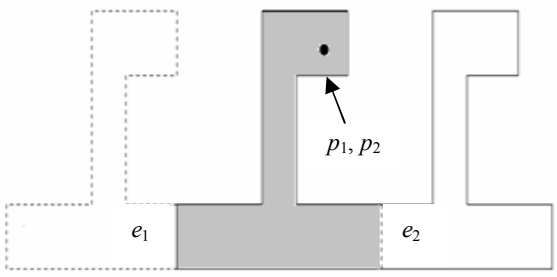

Рис. 3. Внутренние ребра e_1 и e_2

Вычисление пересечения многоугольников требуется в алгоритмах [5] и [6], а также в двух предлагаемых далее алгоритмах. Есть еще несколько общих понятий и связанных с ними алгоритмических действий, которые используются во всех рассматриваемых в данной статье алгоритмах. Это, например, триангуляция простого многоугольника и вычисление скелета многоугольника видимости. Триангуляция многоугольника необходима, по крайне мере, как

вспомогательное действие при вычислении кратчайшего пути из одной точки внутри многоугольника в другую [8], и будет рассмотрена далее при обсуждении алгоритма локализации, основанного на триангуляции. Для обсуждения понятия скелета многоугольника видимости рассмотрим следующие определения, для краткости опуская некоторые детали.

Пусть **P** обозначает границу простого многоугольника P. Две точки в **P** видимы друг из друга, если отрезок, соединяющий их, пересекает **P** только в этих конечных точках. Две точки в P видимы друг из друга, если отрезок, соединяющий их, полностью лежит в P и если пересекает **P**, то только в этих конечных точках. Многоугольник видимости $V(p)$, для любой точки p из P, состоит из всех точек P, которые являются видимыми из p.

Определение. В заданном простом многоугольнике P *ячейкой видимости* $C \in P$ называется максимальное (по включению) связное подмножество P со свойством, что любые две точки в C видят одно и то же подмножество вершин P.

Алгоритм [4] основан на специальной предобработке многоугольника карты – разбиении на ячейки видимости (рис. 4). Сложность этого алгоритма $O(n^5 \log n)$ [4].

Будем говорить, что в многоугольнике видимости $V(p)$ есть *первичные* вершины, т. е. вершины исходного многоугольника P, и есть *вторичные* вершины, т. е. вершины, образованные вследствие рассечения ребер исходного многоугольника. Аналогично ребра многоугольника видимости $V(p)$ можно разделить на три вида: 1) *полное* ребро является ребром исходного многоугольника P, 2) *полуребро* имеет одним концом первичную вершину, а вторым концом вторичную, 3) *частичное* ребро связывает вторичные вершины и является частью ребра исходного многоугольника P.

Определение. Скелет $V^*(p)$ многоугольника видимости $V(p)$ – это многоугольник, образованный первичными вершинами $V(p)$. Скелет многоугольника видимости также можно рассматривать как многоугольник, образованный всеми полными ребрами из $V(p)$ так, что они связаны в последовательность ребер дополняющими искусственными ребрами (рис. 5). Важно, что при этом ребра скелета промаркированы. В [4] показано, что вычисление многоугольника видимости и, следовательно, его скелета имеет сложность $O(n)$.

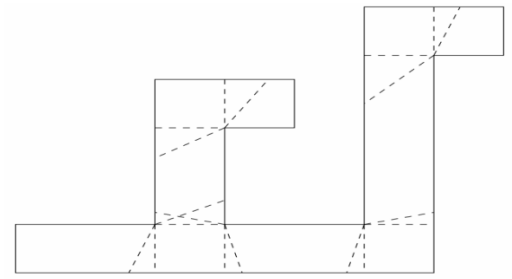

Рис. 4. Разбиение многоугольника P на ячейки видимости

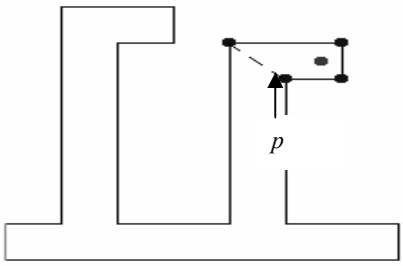

Рис. 5. Скелет $V^*(p)$ многоугольника видимости

2. ПРИБЛИЖЁННЫЕ АЛГОРИТМЫ ЛОКАЛИЗАЦИИ РОБОТА

Дадим описание двух предлагаемых алгоритмов локализации робота. Начнем с алгоритма локализации мобильного робота с использованием триангуляции карты.

Как уже отмечалось, триангуляция многоугольника карты необходима во всех алгоритмах как вспомогательное действие при вычислении кратчайшего пути из одной точки внутри многоугольника в другую. Однако, используя триангуляцию исходного многоугольника как предобработку, во-первых, можно эффективно реализовать и некоторые другие действия, например, построение многоугольника видимости, а во-вторых, можно использовать разбиение карты на множество треугольников для выбора перемещений робота на этапе отсечения гипотез. Например, путь робота может проходить по центрам треугольников или по средним точкам ребер триангуляции.

С учетом данных ранее определений можно описать предлагаемый алгоритм локализации робота с использованием триангуляции карты. Пусть на входе алгоритма заданы многоугольник карты P и робот, помещенный в неизвестное начальное местоположение в P. Алгоритм состоит из следующих действий.

1. Вычислить многоугольник видимости V по данным сенсоров робота в текущем неизвестном начальном местоположении робота.

2. Сгенерировать множество гипотез H на карте P, которые соответствуют полученному многоугольнику видимости V.

3. Выполнить триангуляцию многоугольника карты.

4. Выбрать произвольную гипотезу h_i из H и соответствующую точку гипотетического местоположения как исходную для построения оверлеев.

5. Операции в п. 5–7 осуществляются далее для всех активных (не отклоненных пока) гипотез h_j (j = 1, 2,…, k`).

6. Вычислить для активных гипотез Intersection $(P_1, P_2, ..., P_{k'})$ и связный компонент F, содержащий стартовую точку.

7. Найти точку r в множестве точек на серединах ребер триангуляции и центров треугольников в пределах многоугольника F как такую точку, которая является ближайшей к текущему положению робота среди точек, устраняющих лишние гипотезы.

8. Переместить робота в точку r.

9. Устранить «отклоненные» гипотезы, сравнивая данные о текущем многоугольнике видимости, выдаваемые роботом в точке r, с данными о видимости, вычисленными во всех эквивалентных точках, соответствующих всем активным гипотезам.

10. Пусть E – это множество устраненных гипотез. Повторять шаги 4–8, пока в множестве активных гипотез H–E не останется только одна гипотеза, которая и будет соответствовать истинному начальному местоположению робота.

Сложность этого алгоритма приведена в табл. 1.

Таблица 1

Сложность алгоритма с использованием

Шаг	*Действия*	*Сложность укрупненных действий*
1, 2	*Генерация гипотез*	$O(mn^2)$
3	*Триангуляция многоугольника карты*	$O(n \log^* n)$
5	*Построение оверлейных пересечений относительно выбранной гипотезы, k' – число активных гипотез*	$O(k'n \log n)$
6, 7	*Обследование $3k(n-2)$ точек на ребрах и центрах треугольников для устранения гипотез. Вычисление кратчайших путей для определения ближайшей точки, устраняющей гипотезы*	$k'O(n \log^* n) + 3k'(n-2)O(n) = k'O(n^2)$
8	*Сравнение данных о многоугольниках видимости для активных гипотез и текущего положения робота*	$3k'(n-2)O(n)$
Полная сложность: $O(mn^2) + O(n \log^* n) + \sum_{k'=1}^{k-1}\left[O(k'n \log n) + k'O(n^2) + 3k'(n-2)O(n)\right] = O(n^4)$		

Триангуляцию простого многоугольника можно реализовать с использованием эффективного и практичного алгоритма [9]. Выходом этого алгоритма является множество треугольников, заданных номерами своих вершин. Для дальнейшего использования в алгоритме локализации множество треугольников преобразуется за время $O(n)$ в специальную структуру данных [7]. Это представление, по сути, является одним из адаптированных к триангуляции вариантов реберного списка, используемого для представления планарного подразбиения

плоскости [2]. Каждый треугольник в этой структуре представлен своими тремя вершинами и тремя указателями на соседние треугольники, смежные с ним чрез его стороны. Структура триангуляции получается в два этапа. Сначала по заданному множеству треугольников для каждой вершины триангуляции формируются списки треугольников, в которые входит эта вершина. Затем формируется собственно структура триангуляции, при этом информация о прилегающих треугольниках получается для каждой пары его вершин путем анализа списков, полученных на первом этапе.

Использование структуры триангуляции позволяет эффективно реализовать некоторые базовые операции алгоритма локализации. Например, локализация точки в триангуляции осуществляется с использованием приема кэширования треугольников [7], а систематически используемая операция построения многоугольника видимости и скелета видимости реализована на основе поиска в ширину на графе триангуляции. При этом удается в среднем избежать просмотра всех вершин многоугольника карты, анализируя смежные относительно текущего положения треугольники.

Второй из предлагаемых алгоритмов – это *приближенный алгоритм локализации мобильного робота с использованием окон в многоугольнике карты*. Понятие окна было использовано в работе [6] в алгоритме локализации, сложность которого $O(n^{12})$. Далее предлагается более простой приближенный алгоритм, первоначальный вариант которого был рассмотрен в [10].

Рассмотрим гипотезу h_j ($h_j \neq h_i$), и обозначим как F_{ij} такую грань в $Intersection(P_i, P_j)$, которая содержит начальную позицию γ_0 (см. например, F_{12} на рис. 3). Грань F_{ij} имеет самое большее $2n$ ребер [6].

Каждое из $O(n)$ ребер $e \in F_{ij}$ может быть ребром одного из трех типов: (i) e лежит на границе P_i, но не P_j; (ii) e лежит на границе P_j, но не P_i; или (iii) e лежит на границе и P_i , и P_j. Робот может различить h_i и h_j, если и только если он видит ребро e типа (i) или (ii) (рис. 6).

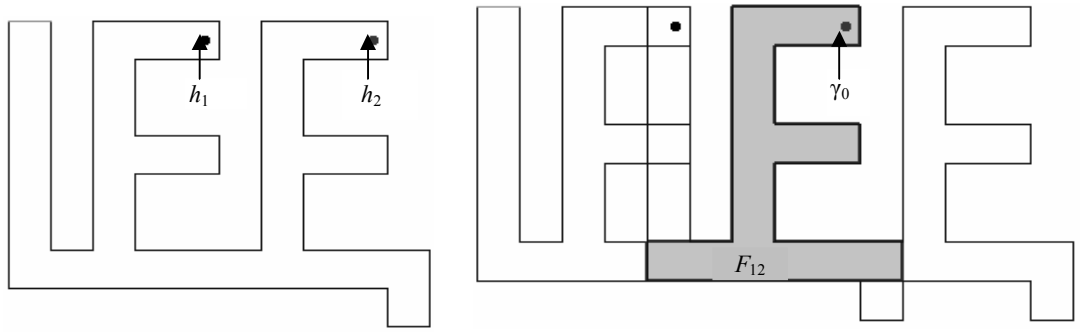

Рис. 6. Слева многоугольник P с двумя гипотезами h_1 и h_2 и справа наложение P_1 и P_2 с выделенной гранью F_{12}, содержащей γ_0

Если из точки γ_0 видно какое-либо ребро типа (i) или типа (ii), то робот может различить h_i и h_j, не перемещаясь из γ_0. Предположим, что все ребра F_{ij}, которые видимы из γ_0, имеют тип (iii). Пусть ребро e грани F_{ij} имеет тип (i) или тип (ii). Множество $VP(e)$ точек F_{ij}, которые видимы с некоторой точки ребра e, является простым многоугольником (многоугольник видимости e) в пределах F_{ij}, который по нашему предположению не включает точку γ_0. В грани F_{ij} существует хорда $w(e)$, которая лежит на границе $VP(e)$ и отделяет e от γ_0. Отрезок $w(e)$ можно назвать *окном* [6] (рис. 7).

С учетом данных определений можно описать предлагаемый алгоритм локализации робота с использованием окон карты. Пусть на входе алгоритма заданы многоугольник карты P и робот, помещенный в неизвестное начальное местоположение в P. Алгоритм состоит из следующих действий:

1. Вычислить многоугольник видимости V по данным сенсоров робота в текущем неизвестном начальном местоположении робота.

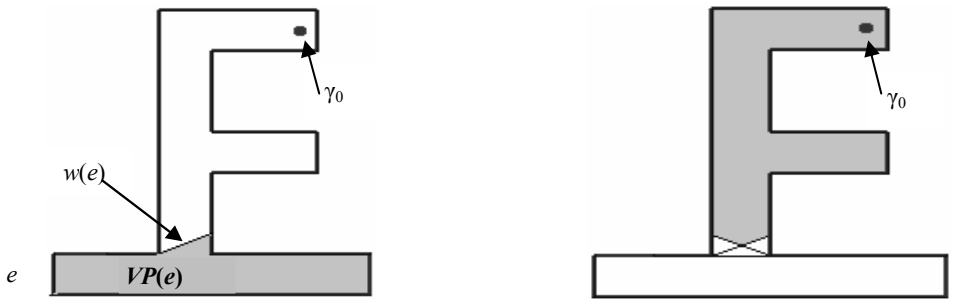

Рис. 7. Слева изображен многоугольник видимости *VP*(*e*) и показаны ребро *e* типа (i) и соответствующая хорда (окно) *w*(*e*). Справа показаны все окна *w*(*e*) для ребер типа (i) или (ii) грани F_{ij}

2. Сгенерировать множество гипотез *H* на карте *P*, которые соответствуют полученному многоугольнику видимости *V*.

3. Выбрать произвольную гипотезу h_i из *H* и соответствующую точку гипотетического местоположения как исходную для построения оверлеев.

Операции в п. 4–7 осуществляются для всех активных (не отклоненных пока) гипотез h_j $(j = 1,2,...,k')$.

4. Построить *Overlay* (P_i, P_j) и *Intersection* (P_i, P_j) экземпляров карты P_i и P_j.

5. Вычислить для *Intersection* (P_i,P_j) связный компонент F_{ij}, содержащий стартовую точку.

6. Построить все окна компонента пересечения F_{ij}.

7. Вычислить средние точки всех окон в F_{ij} и найти среди них такую точку r_j, которая является ближайшей к текущему положению робота, т. е. найти «ближайшее» окно в F_{ij}.

8. Вычислить для активных гипотез *Intersection* $(P_1, P_2, ..., P_{k'})$ и связный компонент *F*, содержащий стартовую точку. Среди тех точек r_j, которые попадают в *F*, найти ближайшую к текущему положению робота точку *r*.

9. Переместить робота в точку *r*.

10. Устранить «отклоненные» гипотезы, сравнивая данные о текущем многоугольнике видимости, выдаваемые роботом в точке *r*, с данными о видимости, вычисленными во всех эквивалентных точках, соответствующих всем активным гипотезам.

11. Пусть *E* – это множество устраненных гипотез. Повторять шаги 3–10, пока в множестве активных гипотез *H*−*E* не останется только одна гипотеза, которая и будет соответствовать истинному начальному местоположению робота.

Сложность этого алгоритма (по шагам и суммарно) приведена в табл. 2.

Таблица 2

Алгоритм локализации работы с использованием окон карты

Шаг	*Действия*	*Сложность укрупненных действий*
1, 2	*Генерация гипотез*	$O(mn^2)$
4–7, 8	*Построение оверлейных пересечений относительно выбранной гипотезы, k' – число активных гипотез*	$O(k'n\log n)$
6, 8	*Построение «окон» связанного компонента оверлейного пересечения*	$O(2k'n^2)$
7, 8	*Обследование $2k'n$ точек на серединах окон. Вычисление кратчайших путей для определения ближайших окон*	$k'O(n\log^* n) + 2k'nO(n) = k'O(n^2)$
10	*Сравнение данных о многоугольниках видимости для активных гипотез и текущего положения робота*	$2k'nO(n)$
Полная сложность: $O(mn^2) + \sum_{k'=1}^{k-1}\left[O(k'n\log n) + O(2k'n^2) + k'O(n^2) + 2k'nO(n)\right] = O(n^4)$		

3. ЭКСПЕРИМЕНТАЛЬНОЕ ИССЛЕДОВАНИЕ АЛГОРИТМОВ

Сравнительные характеристики алгоритма локализации робота с использованием декомпозиции карты на ячейки видимости [4] со сложностью $O(n^5 \log n)$ были проанализированы в [11]. Было установлено экспериментально, что время работы этого алгоритма существенно превышает время работы алгоритма, использующего рандомизацию при проверке гипотез [5], а также время алгоритма локализации робота с использованием триангуляции карты [11]. По этой причине алгоритм, основанный на использовании ячеек видимости, в данном экспериментальном испытании не участвовал. Также из экспериментального исследования исключен алгоритм локализации робота на основе решения полугрупповой задачи Штейнера [6], поскольку он имеет вычислительную сложность $O(n^{12})$ и заведомо уступает другим алгоритмам по времени работы.

Экспериментальное исследование предлагаемых алгоритмов основано на их программной реализации (Visual C++ 2010), а также реализации алгоритма [5] и сравнении таких их характеристик, как величина суммарного пути перемещения робота и время работы алгоритма локализации. При проведении эксперимента использовалась генерация карт различных модельных типов. Пример такой генерации приведен на рис. 8, а фрагмент карты с примером работы алгоритма локализации с использованием триангуляции карты приведен на рис. 9 (слева). Здесь размер карты $n = 746$, а число гипотез $k = 7$. В испытании рандомизированного алгоритма 2 [5] были использованы два варианта для количества случайных точек $X = 100$ и $X = 500$.

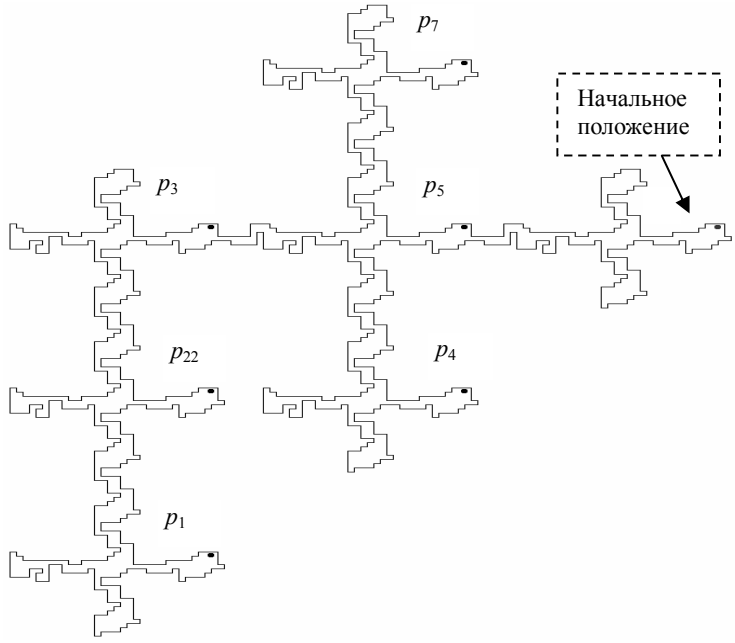

Рис. 8. Пример генерации карты. Размер карты $n = 746$, число гипотез $k = 7$

Можно заметить, что полученные характеристики алгоритмов (табл. 3) существенно зависят от начального расположения робота (от номера гипотезы). По этой причине целесообразно усреднять по гипотезам не сами характеристики, полученные для разных алгоритмов, а их отношения. При этом средние значения длины пути $\overline{d} = \frac{1}{7}\sum_{i=1}^{7} d_i$ и времени работы $\overline{t} = \frac{1}{7}\sum_{i=1}^{7} t_i$ нужны фактически лишь для задания масштаба этих параметров, а средние значения отношений

$$s_i^{(2,1)} = \frac{d_i^{(2)}}{d_i^{(1)}},\ s_i^{(2',1)} = \frac{d_i^{(2')}}{d_i^{(1)}},\ s_i^{(3,1)} = \frac{d_i^{(3)}}{d_i^{(1)}}, s_i^{(2,1)} = \frac{t_i^{(2)}}{t_i^{(1)}},\ s_i^{(2',1)} = \frac{t_i^{(2')}}{t_i^{(1)}},\ \text{и}\ s_i^{(3,1)} = \frac{t_i^{(3)}}{t_i^{(1)}}$$

характеризуют сравнительную эффективность алгоритмов. При этом в качестве «эталона» для сравнения выбраны характеристики алгоритма локализации на основе триангуляции, т. е. в этой «шкале» их значения равны 1. В табл. 4 приведены значения отношений s_i .

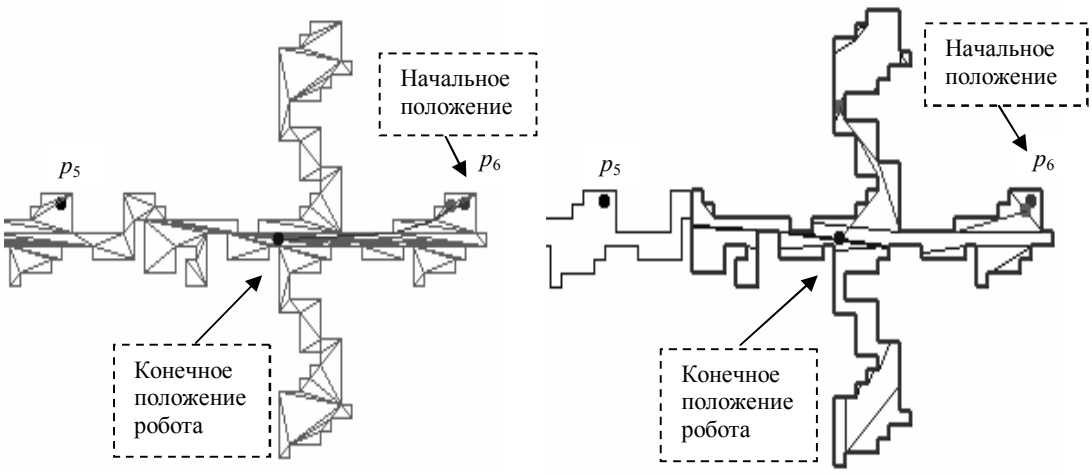

Рис. 9. Фрагмент карты: слева пример работы алгоритма локализации с использованием триангуляции карты, справа – алгоритма локализации с использованием окон карты

Таблица 3

Характеристика алгоритмов

№	*d, отн. ед.*				*t*, с			
	Алгоритм 1	Алгоритм 2		Алгоритм 3	Алгоритм 1	Алгоритм 2		Алгоритм 3
		$X = 100$	$X = 500$			$X = 100$	$X = 500$	
1	464,9	529,2	504,9	500,5	1800,2	788,5	2890,0	273,6
2	303,9	266,0	225,1	220,2	1653,1	683,8	2462,1	230,0
3	90,2	113,5	76,4	69,3	1145,9	499,2	1694,3	178,2
4	461,6	520,1	457,5	499,9	1843,7	808,6	2409,2	278,1
5	91,2	117,0	96,8	70,3	1083,0	459,5	1665,0	178,0
6	92,7	235,9	172,4	175,5	973,8	519,6	2364,6	186,5
7	302,1	245,5	240,4	175,5	1560,9	494,3	2709,4	190,3

Таблица 4

Значения отношений s_i

№	*d, отн. ед.*			*t*, с		
	$s_i^{(2,1)} = \frac{d_i^{(2)}}{d_i^{(1)}}$	$s_i^{(2',1)} = \frac{d_i^{(2')}}{d_i^{(1)}}$	$s_i^{(3,1)} = \frac{d_i^{(3)}}{d_i^{(1)}}$	$s_i^{(2,1)} = \frac{t_i^{(2)}}{t_i^{(1)}}$	$s_i^{(2',1)} = \frac{t_i^{(2')}}{t_i^{(1)}}$	$s_i^{(3,1)} = \frac{t_i^{(3)}}{t_i^{(1)}}$
1	1,14	1,09	1,08	0,44	1,61	0,15
2	0,88	0,74	0,72	0,41	1,49	0,14
3	1,26	0,85	0,77	0,44	1,48	0,16

Окончание табл. 4

№	*d*, отн. ед.			*t*, с		
	$s_i^{(2,1)}=\frac{d_i^{(2)}}{d_i^{(1)}}$	$s_i^{(2',1)}=\frac{d_i^{(2')}}{d_i^{(1)}}$	$s_i^{(3,1)}=\frac{d_i^{(3)}}{d_i^{(1)}}$	$s_i^{(2,1)}=\frac{t_i^{(2)}}{t_i^{(1)}}$	$s_i^{(2',1)}=\frac{t_i^{(2')}}{t_i^{(1)}}$	$s_i^{(3,1)}=\frac{t_i^{(3)}}{t_i^{(1)}}$
4	1,13	0,99	1,08	0,44	1,31	0,15
5	1,28	1,06	0,77	0,42	1,54	0,16
6	2,54	1,86	1,89	0,53	2,43	0,19
7	0,81	0,80	0,58	0,32	1,74	0,12
$\overline{s}^{(l,1)}=\frac{1}{7}\sum_{j=1}^{7}s_i^{(l,1)}$	1,29	1,05	0,99	0,43	1,65	0,15

На рис. 10 и 11 приведены графики зависимостей средних значений указанных отношений от размера многоугольника карты. Различные типы линий графиков на рис. 10 и 11 соответствуют отношениям: 1 – отношению $\overline{s}^{(2,1)}$; 2 – отношению $\overline{s}^{(2',1)}$; 3 – отношению $\overline{s}^{(3,1)}$. Сравнительный анализ данных, соответствующих рис. 10 и 11, показывает следующее:

1. Значения среднего пути робота для алгоритмов 1, 2′ при $X = 500$ и 3 очень близки, а для алгоритма 2 при $X = 100$ несколько больше. Например, для случая $n = 746$, соответствующего табл.1, имеем

$$\overline{d}_1 = 258,09;\ \overline{d}_2/\overline{d}_1 = 1,12;\ \overline{d}_{2'}/\overline{d}_1 = 0,98;\ \overline{d}_3/\overline{d}_1 = 0,94;$$

где нижние индексы соответствуют номерам алгоритмов, но для алгоритма 2 использован индекс 2 при $X = 100$ и индекс 2′ при $X = 500$.

2. Самое большое время показывает алгоритм 2′. При $n = 746$, например,

$$\overline{t}_1 = 1437,2 \text{ с; а } \overline{t}_2/\overline{t}_1 = 0,42;\ \overline{t}_{2'}/\overline{t}_1 = 1,61;\ \overline{t}_3/\overline{t}_1 = 0,15,$$

т. е. алгоритмы 1, 2 и 3 выполняются существенно быстрее. Лучшим по времени оказывается алгоритм локализации робота с использованием окон карты (при $n = 746$ он в среднем примерно в 6 раз быстрее, чем алгоритм на основе триангуляции). Характеристики рандомизированного алгоритма существенно зависят от параметра X, например, при $X = 100$ он работает быстрее триангуляционного алгоритма, но менее точен, а при $X = 500$ он более точен, но работает медленнее триангуляционного алгоритма (при $n = 746$ в среднем примерно в 2.5 раза).

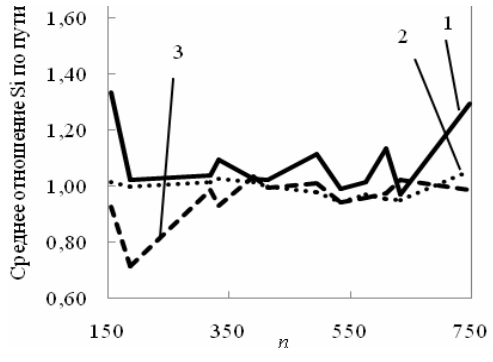

Рис. 10. Зависимость средних значений для отношений длин путей

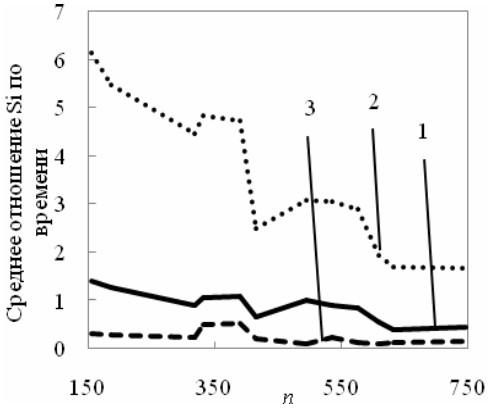

Рис. 11. Зависимость средних значений для отношений времени работы алгоритмов

ЗАКЛЮЧЕНИЕ

Проведенные эксперименты с другими модельными конфигурациями карты показали аналогичные результаты и позволяют сделать вывод, что все рассмотренные приближенные алгоритмы обеспечивают сравнимую точность, но время работы алгоритма локализации робота с использованием окон карты меньше, чем у других алгоритмов. Близкие по времени результаты показывает и алгоритм на основе триангуляции.

Однако время работы и асимптотическая сложность $O(n^4)$ предложенных алгоритмов для многих практических задач все-таки велики. Их уменьшения можно ожидать за счет оптимизации наиболее трудоемких шагов алгоритма, а также за счет их реализации на параллельных структурах, например, с использованием графических ускорителей и гетерогенных вычислительных структур [13], [14]. Такая перспектива позволяет рассматривать оба предложенных алгоритма, поскольку, хотя алгоритм на основе триангуляции проигрывает в быстроте алгоритму с использованием окон карты, его параллельная реализация может оказаться более эффективной. Это соображение основано на наличии эффективного параллельного алгоритма триангуляции [13]. Следует отметить, что в исследованной реализации алгоритма с использованием окон карты для пересечения экземпляров карты использован модифицированный расширенный алгоритм Грейнера–Хорманна [15]. Для этих же целей может быть использован алгоритм построения пересечения, основанный на триангуляции, например [16]. Параллельные реализации алгоритма Грейнера–Хорманна в [15] и [17] показали ускорение работы в среднем в 3–8 раз по сравнению с последовательной реализацией. Это позволяет надеяться и на эффективность планируемых параллельных реализаций обоих предложенных алгоритмов локализации робота.

СПИСОК ЛИТЕРАТУРЫ

[1] **Dudek G., Jenkin M.** Computational principles of mobile robotics. – 2nd rev. ed. – Cambrigde Univ. Press, 2010. – 406 p.

[2] **De Berg M., Cheong O., Van Kreveld M., Overmars M.** Computational geometry: algorithms and applications. – 3nd rev. ed. – Berlin; Heidelberg: Springer-Verlag, 2008. – 386 p.

[3] **Guibas L.J., Motwani R., Raghavan P.** The robot localization problem // The SIAM J. on Computing. – 1997. – № 26. – P. 1120–1138.

[4] **Dudek G., Romanik K., Whitesides S.** Localizing a robot with minimum travel // The SIAM J. on Computing. – 1998. – **№** 27. – P. 583–604.

[5] **Rao M., Dudek G., Whitesides S.** Randomized algorithms for minimum distance localization // Intern. J. Robotics Research. – 2007. – № 26. – P. 917–934.

[6] **Koenig S., Mitchell J. S. B., Mudgal A., Tovey C.** A near-tight approximation algorithm for the robot localization problem // The SIAM J. on Computing. – 2009. – **№** 39. – P. 461–490.

[7] **Скворцов А.В.** Триангуляция Делоне и её применение. – Томск: Изд-во Том. гос. ун-та, 2002. – 128 с.

[8] **Hershberger J. and Snoeyink J.** Computing minimum length paths of a given homotopy class // Computational Geometry. Theory and Applications. – 1994. – № 4. – P. 63–98.

[9] **Seidel R.** A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons // Computational Geometry. Theory and Applications. – 1991. – № 1 (1). – P. 51–54.

[10] **Дао Зуй Нам, Ивановский С.А.** Приближенный алгоритм локализации мобильного робота с использованием окон в многоугольнике карты // Известия СПБГЭТУ «ЛЭТИ». – 2014. – № 3. – С. 38–43.

[11] **Дао Зуй Нам, Ивановский С.А.** Экспериментальный анализ алгоритмов локализации мобильного робота // Известия СПБГЭТУ «ЛЭТИ». – 2014. – № 1. – С. 19–24.

[12] **Dao Duy Nam, Ivanovskiy S.A.** Two new approaches to minimum distance localization // J. of Science and Technology Univ. of Danang. – 2013. – № 12 (73). – P. 52–57.

[13] **Qi M., Cao T.-T., Tan T.S.** Computing 2D constrained delaunay triangulation using graphics Hardware [Electronic resource] // Technical Report / National University of Singapore, School of Computing. – 2011. – # TRB3/11. – 9 p. – URL: http://www.comp.nus.edu.sg/~tants/cdt.html

[14] **Сандерс Дж., Кэндрот Э.** Технология CUDA в примерах: введение в программирование графических процессоров. – М.: ДМК Пресс, 2011. – 232 с.

[15] **Фирсов М.А., Ивановский С.А.** Параллельная реализация алгоритма построения пересечения простых полигонов с использованием технологии CUDA // Известия СПБГЭТУ «ЛЭТИ». – 2013. – № 9. – С. 29–34.

[16] **Скворцов А.В.** Построение объединения, пересечения и разности произвольных многоугольников в среднем за линейное время с помощью триангуляции // Вычисл. методы и программирование. – 2002. – Т. 3. – С. 116–123.

[17] **Фирсов М.А.** Сравнение параллельных реализаций алгоритма пересечения полигонов на многоядерном и на графическом процессорах // 67-я научно-техническая конференция профессорско-преподавательского состава университета, Санкт-Петербург, 27 января–3 февраля 2014 г.: сб. докл. студентов, аспирантов и молодых ученых. – СПб.: Изд-во СПБГЭТУ «ЛЭТИ», 2014. – С. 103–107.

Дао Зуй Нам, аспирант Санкт-Петербургского государственного электротехнического университета «ЛЭТИ» им. В.И. Ульянова (Ленина). Основное направление научных исследований – вычислительная геометрия, визуализация алгоритмов. Имеет более 5 публикаций. E-mail: duynam81@mail.ru

Ивановский Сергей Алексеевич, кандидат технических наук, доцент Санкт-Петербургского государственного электротехнического университета «ЛЭТИ» им. В.И. Ульянова (Ленина). Основное направление научных исследований – построение и анализ алгоритмов, вычислительная геометрия, визуализация алгоритмов, динамическая сегментация изображений, формальные методы верификации программ. Имеет более 100 публикаций. E-mail: saivanovsky@mail.ru

Approximation algorithms for the mobile robot localization[*]

DAO DUY NAM[1], S.A. IVANOVSKIY[2]

[1] Saint Petersburg Electrotechnical University "LETI", 5, Professor Popov St., St. Petersburg, 197376, Russian Federation, post-graduate student, e-mail: duynam81@mail.ru
[2] Saint Petersburg Electrotechnical University "LETI", 5, Professor Popov St., St. Petersburg, 197376, Russian Federation, doctor of philosophy, associate professor, e-mail: saivanovsky @mail.ru

Two approximation algorithms of localization of a mobile robot supplied with the map in the form of a simple polygon without holes are considered. The localization hypotheses have corresponding map copies with estimated positions of the robot marked on them. The robot has to determine its true initial location moving and surveying a visible vicinity to eliminate all false hypotheses. Thus the total length of movements of the robot has to be minimal. The optimizing problem of robot localization is NP-complete therefore approximation algorithms are considered. One of the algorithms is based on the use of the triangulation of a simple polygon representing the map. Preprocessing in the form of the triangulation of a simple polygon makes it possible to realize effectively the main algorithm actions such as calculating a visibility polygon, finding the shortest path between two points in a polygon, and eliminating false hypotheses. The second algorithm uses overlay (intersection) of map copies. The so-called windows are allocated in the intersection and "looking" into them the robot eliminates false hypotheses. Based on the program realization of several algorithms they have been experimentally studied using the generated model map. Numerical results of computer experiments and their interpretation are given. The proposed algorithms are only slightly inferior to the algorithms known earlier by the minimization criterion of the length of the way passed by the robot, but in model examples work quicker. Possible ways of reducing the algorithm operating time due to using parallelism are analyzed.

Keywords: Computational geometry, robotics, mobile robot, robot localization, simple polygon, visibility polygon, hypothesis generation, hypothesis elimination, polygon overlay, polygon triangulation, algorithm complexity, approximation algorithm, map generation, experimental algorithm study

REFERENCES

[1] Dudek G., Jenkin M. Computational Principles of Mobile Robotics. 2nd rev. ed. Cambrigde Univ. Press, 2010. 406 p.

[2] De Berg M., Cheong O., Van Kreveld M., Overmars M. Computational Geometry: Algorithms and Applications. 3nd rev. ed., Berlin, Heidelberg, Springer-Verlag, 2008. 386 p.

[3] Guibas L.J., Motwani R., Raghavan P. The robot localization problem. The SIAM J. on Computing, 1997, vol. 26, pp. 1120-1138.

[4] Dudek G., Romanik K., Whitesides S. Localizing a robot with minimum travel. The SIAM J. on Computing, 1998, vol. 27, pp. 583-604.

[5] Rao M., Dudek G., Whitesides S. Randomized algorithms for minimum distance localization. Intern. J. Robotics Research, 2007, vol. 26, pp. 917-934.

[6] Koenig S., Mitchell J. S. B., Mudgal A., Tovey C. A near-tight approximation algorithm for the robot localization problem. The SIAM J. on Computing, 2009, vol. 39, pp. 461-490.

[7] Skvortsov A.V. *Trianguliatsiia Delone i ee primenenie* [Delaunay triangulation and its application]. Tomsk, Tomsk State University Publ., 2002. 128 p.

[8] Hershberger J. and Snoeyink J. Computing minimum length paths of a given homotopy class. Computational Geometry. Theory and Applications, 1994, no. 4, pp. 63-98.

[9] Seidel R. A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. Computational Geometry. Theory and Applications, 1991, no. 1 (1), pp. 51-54.

[10] Dao Duy Nam, Ivanovskiy S.A. Priblizhennyi algoritm lokalizatsii mobil'nogo robota s ispol'zovaniem okon v mnogougol'nike karty [Approximation algorithm localization mobile robot with using windows in polygon map]. *Isvestia SPBSETU «LETI»* [Bulletin of the Saint Petersburg Electrotechnical University «LETI»], 2014, no. 3, pp. 38-43.

[*] *Manuscript received December 24, 2014.*

[11] Dao Duy Nam, Ivanovskiy S.A. Eksperimental'nyi analiz algoritmov lokalizatsii mobil'nogo robota [Experimental analysis of algorithms for localizing a mobile robot]. *Isvestia SPBSETU «LETI»* [Bulletin of the Saint Petersburg Electrotechnical University «LETI»], 2014, no. 1, pp. 19-24.

[12] Dao Duy Nam, Ivanovskiy S.A. Two new approaches to minimum distance localization. J. of Science and Technology Univ. of Danang, 2013, no. 12 (73), pp. 52-57.

[13] Qi M., Cao T.-T., Tan T.S. Computing 2D Constrained Delaunay Triangulation Using Graphics Hardware. Technical Report. National University of Singapore, School of Computing, March 2011, # TRB3/11, 9 p. Available at: http://www.comp.nus.edu.sg/~tants/cdt.html

[14] Sanders J., Kandrot Ed. CUDA by Example: Introduction to general-purpose GPU programming. Moscow, DMK Publ., 2011. 232 p.

[15] Firsov M.A., Ivanovskiy S.A. Parallel'naia realizatsiia algoritma postroeniia peresecheniia prostykh poligonov s ispol'zovaniem tekhnologii CUDA [Parallel implementation of algorithm for construction of simple plane polygon intersection with use of CUDA technology]. *Isvestia SPBSETU «LETI»* [Bulletin of the Saint Petersburg Electrotechnical University «LETI»], 2013, no. 9, pp. 29-34.

[16] Skvorsov A.V. Postroenie ob"edineniia, peresecheniia i raznosti proizvol'nykh mnogougol'nikov v srednem za lineinoe vremia s pomoshch'iu trianguliatsii [Creation combination, intersection and difference of any polygons on the average in linear time with using triangulation]. *Vychislitel'nye metody i programmirovanie – Computing methods and programming*, 2002, vol. 3, pp. 116-123.

[17] Firsov M.A. Sravnenie parallel'nykh realizatsii algoritma peresecheniia poligonov na mnogoiadernom i na graficheskom protsessorakh [Comparison of parallel implementations of the algorithm intersection polygons on a multicore]. *67-ia nauchno-tekhnicheskaia konferentsiia professorsko-prepodavatel'skogo sostava universiteta, Sankt-Peterburg, 27 ianvaria–3 fevralia 2014 g.: sb. dokl. studentov, aspirantov i molodykh uchenykh* [67-th scientific conference of the faculty of the University, Saint-Petersburg, January 27-February 3, 2014, Collection of reports of students, postgraduates and young scientists], Saint Petersburg, ETU «LETI» Publ., 2014, pp. 103-107.