К.С. Яковлев, Д.А. Макаров, Е.С. Баскин

Метод автоматического планирования траектории беспилотного летательного аппарата в условиях ограничений на динамику полета[1]

Аннотация. В статье решается задача планирования траектории беспилотного летательного аппарата мультироторного типа. Предлагается метод, позволяющий на основе модели динамики полета и допустимого управления, а также при некоторых разумных предположениях, определить ограничения на геометрию траектории. На основе этого описывается новый алгоритм планирования траектории, учитывающий геометрические ограничения. Приводятся результаты экспериментальных исследований алгоритма. В качестве модельной рассматривается задача планирования траектории маловысотного полета в городских условиях.

Ключевые слова: беспилотное транспортное средство, беспилотный летательный аппарат, система управления, интеллектуальная система управления, навигация, планирование, планирование траектории, построение пути, эвристический поиск, геометрические ограничения.

Введение

В настоящее время все большее распространение получают малые беспилотные летательные аппараты (БЛА) вертикального взлета и посадки мультироторного типа (мультикоптеры). Это объясняется относительной дешевизной и распространенностью компонентов (роторов, аккумуляторных батарей, микроконтроллеров), а также наличием готовых платформ, таких как AscTec Hummingbird, Parrot Ar.Drone, Mikrokopter, ARDUCopter и т.д. Мультикоптеры достаточно активно используются как в коммерческих (аэрофотосъемка, экологический и сельскохозяйственный мониторинг и др.), так и в научных целях [1]. Основной целью, преследуемой исследователями и разработчиками в области беспилотной техники, является повышение степени автономности объекта управления [2]. Здесь в настоящее время достигнут определенный прогресс [3]. Так, отдельные задачи автоматизации управления мультикоптерами можно считать успешно решенными. К ним относятся, например, задачи автоматического выполнения некоторых основных режимов полета (висение, полет с постоянной скоростью и пр.) и маневров (взлет, посадка, зависание и пр.) и другие.

Многие мультироторные платформы уже включают в состав своих бортовых вычислителей программно- и аппаратно-реализованные средства автоматизации этих маневров/режимов, а также открытый программный интерфейс доступа к ним. Исследователь и разработчик может передавать на борт (обычно по радиоканалу либо по wifi) команды для исполнения («лететь вперед», «набрать высоту», «зависнуть в точке») и быть уверенным в их отработке [4]. Таким образом, на первый план выходят более высокоуровневые задачи: картирование (построение и уточнение в реальном времени карты местности), локализация, планирование траектории и др. Именно вопросам, связанным с решением этих задач, а более точ-

[1]Исследование выполнено за счет гранта Российского научного фонда (проект №14-11-00692).

но – с задачей планирования траектории по имеющейся карте (модели) местности и посвящена данная работа.

Традиционно в искусственном интеллекте задача планирования траектории для некоторого агента рассматривается как задача поиска пути на графе (при этом сам агент обычно считается материальной точкой). Вершинам графа соответствуют различные положения агента в пространстве, ребрам – элементарные траектории, т.е. такие траектории, следование агента по которым считается тривиальной задачей (обычно это отрезки прямых или кривых определенного радиуса). Таким образом, задача планирования траектории сводится к двум подзадачам: построению графа, моделирующего окружающую среду агента, и поиску пути на этом графе.

В качестве графовых моделей окружающей среды обычно используются графы регулярной декомпозиции (ГРД), в англоязычной терминологии – grids [5], схожие с ними модели: MT-графы [6], графы видимости [7], вероятностные схемы местности [8] и др. Достаточно полный и информативный обзор графовых моделей, применяемых для решения задачи планирования траектории на плоскости, приведен в [9].

Методы поиска пути на графе обычно базируются на итерационном обходе вершин графа – принципе, описанном еще в 1959г. Э. Дейкстрой в работе [10]. Зачастую используются всевозможные эвристики (обычно основанные на геометрических свойствах рабочей области и имеющие простую геометрическую интерпретацию) для сокращения пространства поиска [11]. Алгоритмы, реализующие эвристический поиск, являются модификациями (порою весьма значительными) известного в литературе по искусственному интеллекту алгоритма A* [12]. С обзором основных алгоритмов эвристического поиска можно ознакомиться в [13]. Достаточно полный обзор эвристических алгоритмов, применимых для задачи поиска пути на плоскости, приведен на вебресурсе [14] (содержащем, в том числе, ссылки на оригинальные работы). Известны также декомпозиционные алгоритмы поиска пути на графе, основанные на разбиении (возможно рекурсивном) исходной задачи на ряд потенциально легкоразрешимых подзадач (локальных задач) с последующей декомпозицией получившихся локальных решений в искомое. К таким алгоритмам относятся, например, R* [15], HGA* [16] и др. Зачастую декомпозиционные алгоритмы превосходят итерационные по эффективности использования вычислительных ресурсов, но уступают (незначительно) по качеству отыскиваемых решений.

Основная трудность применения известных в искусственном интеллекте алгоритмов эвристического поиска для решения задачи планирования траектории БЛА мультироторного типа заключается в том, что агент (мультикоптер) не может более считаться материальной точкой, динамикой и ориентацией в пространстве которого можно пренебречь. Таким образом, возникает необходимость учитывать, помимо размеров, пространственную ориентацию, возможные ограничения на управление и фазовые координаты, а также динамику полета мультикоптера.

Наиболее простой задачей, имеющей стандартное решение, является учет размера агента при планировании [17]. Гораздо менее проработанными (в виду большей сложности) являются вопросы учета кинематических и динамических ограничений объекта управления.

Известны подходы, когда ограничения на динамику движения агента косвенно учитываются на этапе построения графовой модели – каждая вершина графа содержит не только информацию о положении объекта в пространстве, но и о его ориентации (обычно дискретизированной до нескольких десятков значений). При этом вопрос об определении ребер графа остается открытым – обычно исследователями определяется набор простейших правил перехода, запрещающих «резко» менять пространственную ориентацию агента [18]. Вопрос об исполнимости траектории в общем случае (т.е. вопрос о том, сможет ли объект управления, пусть даже в режиме моделирования, проследовать по траектории, соответствующей найденному на графе пути) при таком подходе остается открытым. Стоит также заметить, что поиск пути на подобных графах весьма трудоемок, так как пространство состояний велико (на порядок больше, чем, если бы граф содержал лишь информацию о положении объекта в пространстве).

Другим распространенным подходом к учету указанных ограничений является подход, основанный на динамическом, итерационном построении графа (совмещенным с поиском пути на нем) путем стохастического моделирования [19]. При этом выбор вершин, потенци-

ально смежных с текущей зачастую осуществляется квазислучайно, а проверка на смежность осуществляется путем моделирования следования по соответствующей траектории. Такая проверка – весьма трудоемкий процесс, вычислительная сложность которого напрямую зависит от сложности модели объекта. Именно поэтому подобный подход обычно применяется для объектов, обладающих достаточно простой динамикой (моделью движения) – в основном для колесных беспилотных транспортных средств, многозвенных манипуляторов и др. [20-22], а не для БЛА мультироторного типа, которые могут обладать весьма сложной нелинейной моделью.

Обобщая можно сделать вывод о том, что весьма актуальной в настоящее время является задача разработки нового метода планирования траектории полета БЛА мультироторного типа, сочетающего в себе особенности обоих указанных подходов и лишенного их характерных недостатков. Необходимо, чтобы на этапе построения графа и поиска пути на нем не осуществлялось ресурсоемкое моделирование полета, при этом построенный путь должен быть реализуем при некоторых допустимых условиях (заданных режиме полета, ветровых нагрузках и т.д.), т.е. должен учитывать ограничения на динамику движения объекта управления, задаваемые моделью. В работе предлагается такой метод, описывается алгоритм его реализующий, проводится экспериментальное исследование алгоритма, показывается его потенциальная применимость на практике для решения определенного класса задач.

1. Постановка задачи

Здесь и далее в качестве модельной будет рассматриваться задача планирования траектории БЛА вертикального взлета и посадки мультироторного типа, осуществляющего маловысотный полет в городских условиях. Предполагается, что полет осуществляется в одной (горизонтальной) плоскости ниже уровня высотных строений (домов), при этом в качестве исходных данных для построения траектории выступает цифровая карта местности, содержащая информацию об этих строениях. Также предполагается известной (заданной) модель динамики полета БЛА, с использованием которой можно моделировать полет и, в частности, определять, является ли та или иная траектория полета (при заданных начальных условиях) реализуемой или нет.

Будем считать, что каждое строение (препятствие) представлено в виде многоугольника, координаты вершин которого известны. Такой способ описания строений достаточно распространен во многих геоинформационных базах данных, например, в открытой цифровой модели местности OSM [23], которую и будем считать основным источником исходных данных о среде функционирования агента (БЛА мультироторного типа).

Известны текущее и целевое положения БЛА в пространстве, привязанные к имеющейся карте местности. Задана скорость, с которой необходимо осуществлять полет. Будем полагать, что БЛА уже имеет необходимую скорость в начальной точке. Задача планирования траектории состоит в описании двумерной кривой, привязанной к имеющейся модели местности, соединяющей начальное и целевое положение БЛА, а также удовлетворяющей ряду условий, о которых будет сказано ниже.

Во-первых, любая точка кривой должна быть проходима для БЛА (т.е. БЛА может находиться в этой точке, не «задевая» препятствия, а желательно – находясь на некотором безопасном расстоянии от них). Таким образом, возникает необходимость учитывать ограничения, связанные с физическими размерами объекта управления и его пространственной ориентацией. Во-вторых, при построении траектории должны учитываться ограничения, накладываемые на динамику полета БЛА. Другими словами, искомая траектория должна быть реализуема в заданной модели динамики полета.

Отдельно необходимо сказать об ограничениях на пространственную ориентацию БЛА в начальный и конечный моменты движения. Заметим, что мультироторный БЛА может разворачиваться в горизонтальной плоскости так, что точка его центра масс остается на месте (если пренебречь влиянием ветра и прочими возмущающими воздействиями). Таким образом, можно считать, что ориентация БЛА в начальном (целевом) положении не является ограничением, которое необходимо учитывать при решении задачи планирования траектории. Однако подчеркнем, что ограничения на пространственную ориентацию во время полета хотя бы в некоторых точках траектории, учи-

тывать необходимо. Об этом будет сказано в разделе 2.1.

Ниже приведем математическую формулировку поставленной задачи. Пусть в двумерном евклидовом пространстве в декартовой системе координат задана прямоугольная область U: $x_{\min} \le x \le x_{\max}$, $y_{\min} \le y \le y_{\max}$ (рабочая область), представимая в виде непересекающегося объединения двух подобластей U_{free} (свободное пространство) и U_{obs} (препятствия). $U_{obs}=\{obs_1, obs_2, \ldots, obs_i, \ldots, obs_N\}$, где $obs_i=\{p_{i1}, p_{i2}, \ldots, p_{ij}, \ldots p_{iMi}\}$ – i-е препятствие, представляющее собой многоугольник, заданный множеством своих вершин $p_{ij}=(x_{ij}, y_{ij})\in U$. Заданы также две точки $s\in U_{free}$ и $g\in U_{free}$ – начальная и целевая соответственно. Объект управления (мультикоптер) представляет собой круг определенного радиуса r. Известна также модель динамики мультикоптера в виде системы дифференциальных уравнений, которая имеет вид:

$$\frac{dx}{dt} = f(x,u),$$

где $f(x,u)$ – вектор функция, $x \in R^n$ – вектор фазовых координат, $u \in R^r$ – вектор управления, t – время.

В нашем случае, подробно описанном ниже, модель задает уравнения поступательного движения центра тяжести летательного аппарата в разных системах координат. Вектор состояния x считается известным. Задано также ограничение на управление – максимальный вектор тяги.

Необходимо построить (описать в каком-либо виде) двумерную кривую tr, соединяющую точки s и g, такую что:

- каждая точка этой кривой является проходимой и лежит в U;
- круг заданного радиуса r с центром в любой точке кривой содержит лишь проходимые точки, т.е. лежит полностью в U_{free};
- траектория является реализуемой с учетом имеющихся ограничений модели, т.е. существует допустимое управление, вдоль которого решение уравнений пространственного перемещения не покидает заданную допустимую окрестность траектории tr.

2. Метод решения

Основная идея предлагаемого метода заключается в декомпозиции решения поставленной задачи на две подзадачи: собственно планирования и определения ограничений на допустимую траекторию движения.

Предлагается следующая модель ограничений. Считается, что реализуемая траектория (для заданного режима полета) представляет собой последовательность отрезков таких, что угол отклонения между любыми смежными отрезками последовательности не превышает (по модулю) некоторого фиксированного значения $\boldsymbol{\alpha}$. Предполагается, что соблюдение этого условия гарантирует реализуемость полученной траектории, т.е. возможность выработки допустимых управляющих сигналов для следования по ней. Для определения таких ограничений предложен метод, основанный на анализе области достижимости динамических систем.

Задача планирования траектории рассматривается как задача поиска пути на графе. Для ее решения используется простая графовая модель, сложность построения которой для рабочей области минимальна. При этом на этапе построения графа учитываются ограничения на размер объекта управления, на этапе поиска пути в графе – имеющиеся геометрические ограничения (ограничения на максимальный угол поворота).

2.1. Метод определения геометрических ограничений

Для определения геометрических ограничений на максимально допустимый угол $\boldsymbol{\alpha}$ осуществим анализ области достижимости динамической системы. Достаточно точное решение этой задачи, как правило, требует больших вычислительных затрат. Здесь ограничимся упрощенным подходом, основанным на некоторых правдоподобных допущениях. Предложенный подход обладает достаточной общностью применения, однако для определенности мы зададимся конкретным типом мультикоптера – квадрокоптером AscTec Hummingbird [24].

Основное наше предположение состоит в том, что условия полета и точность управления с учетом ограничений таковы, что существует возможность построения допустимого управления, которое гарантирует нахождение мультикоптера в некоторой допустимой окрестности желаемой *прямолинейной* траектории полета. Эту окрестность мы будем задавать «трубкой» с радиусом r_d (Рис. 1).

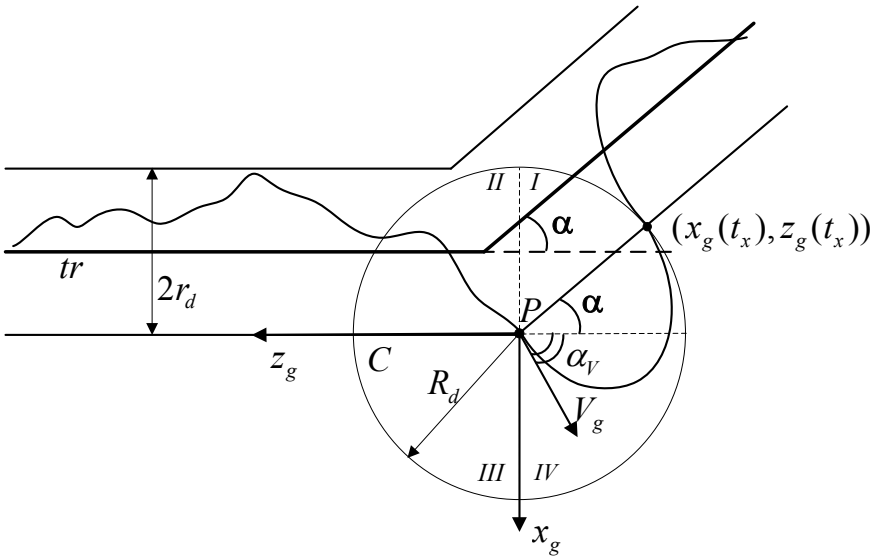

Рис. 1. Метод определения геометрических ограничений

Такое допущение естественно, так как точное следование по заданной траектории в общем случае невозможно даже при полной модели объекта управления и при отсутствии внешних возмущающих факторов. Наличие ограничений на управление, внешних возмущений (порывы ветра и д.р.), ошибок модели и управления еще более усугубляют проблему следования по пути. Конкретное значение r_d зависит от типа БЛА, режима и условий полета и пр. Далее будем полагать, что значение r_d нам известно (например, исходя из опыта эксплуатации выбранного аппарата).

Очевидно, что для представленного на Рис. 1 случая «излома» траектории, наихудшим для управления является положение БЛА в точке *P*. Рассмотрим дополнительную область, определяемую кругом *C* с заданным радиусом R_d с центром в точке *P,* не принадлежащей «трубке» . Будем считать, что ни одна точка в *C* не содержит препятствий. Теперь задача определения геометрических ограничений может быть сформулирована следующим образом. Необходимо найти такой максимальный угол **α**, при котором траектория полета мультикоптера, не покидая круга *C*, вернется опять в допустимую «трубку», которую уже не покинет, так как по допущению выше всегда существует допустимое управление, гарантирующее это. Радиус R_d определяет некоторую область маневра поворота, в которой не должны находиться препятствия. Далее будем считать, что значение R_d задано.

Сделаем дополнительные предположения:

- положим, что в точке *P* в наихудшем случае, который и будем рассматривать, вектор земной скорости аппарата V_g отклонен на угол α_v относительно желаемой траектории полета до текущей точки ее изменения (Рис.1). Траектория задается в земной прямоугольной правой системе координат $Px_gy_gz_g$. Вектор $V_g = V+W$, где *V* – воздушная скорость полета ЛА, *W* – скорость ветра относительно $Px_gy_gz_g$;
- воздушную скорость *V* и массу *m* БЛА будем считать постоянными;
- будем считать, что задана максимальная скорость ветра *W* и в наихудшем случае ветер в каждый момент времени будет стремиться «вытолкнуть» аппарат из круга *C*. Поскольку вектор V_g при криволинейном полете направлен по касательной к траектории, то последнее приводит аппарат к некоторой заданной постоянной скорости V_g, которая и отражает наихудший случай воздействия ветра;
- пусть органы управления в момент присутствия БЛА в точке *P* сразу находятся в нужном положении и их воздействие максимально, а БЛА сразу имеет максимальные необходимые угловые скорости. Иными словами отсутствует переходной процесс в управлении, и полет является сбалансированным и криволинейным. Для выбранной нами модели летательного аппарата такое предположение вполне допустимо, так как AscTec Hummingbird имеет крайне

малый отклик на управляющее воздействие (постоянная времени порядка 0,02 с);

- будем считать, что БЛА совершает плоский поворот без скольжения (вектор скорости лежит в горизонтальной плоскости и совпадает с продольной конструктивной осью БЛА), используя максимальную тягу двигателей F_{max}.

Пусть оси Ox_g, Oz_g земной системы координат $Px_gy_gz_g$ направлены так, как представлено на Рис 1, а ось Oy_g направлена вертикально вверх, тогда воспользуемся следующим алгоритмом нахождения **α**:

1. Проведем численное моделирование движения БЛА, начинающегося в точке *P*, до момента времени t_x, при котором:

1.1. траектория движения пересекает соответствующую *C* окружность, тогда переходим к пункту 2 алгоритма;

1.2. или имеют место неравенства:

$$\frac{-z_g(t_x)}{\sqrt{x_g(t_x)^2+z_g(t_x)^2}}<0,\frac{-x_g(t_x)}{\sqrt{x_g(t_x)^2+z_g(t_x)^2}}\le 0,$$

где $x_g(t_x),z_g(t_x)$ – координаты ЛА в $Ox_gy_gz_g$ в момент времени t_x, тогда переходим к пункту 3 алгоритма;

1.3. или достигнуто максимальное время моделирования, тогда переходим к пункту 3 алгоритма.

2. Вычислить искомый угол по формуле:

$$\boldsymbol{\alpha}(t_x)=$$
$$=\arccos\left(\frac{-z_g(t_x)}{\sqrt{x_g(t_x)^2+z_g(t_x)^2}}\right)*\mathrm{sgn}\left(\frac{-x_g(t_x)}{\sqrt{x_g(t_x)^2+z_g(t_x)^2}}\right);$$

конец алгоритма.

3.Ограничения на геометрию отсутствуют; конец алгоритма.

Условие 1.2 соответствует случаю, когда $180^\circ\le\boldsymbol{\alpha}\le 270^\circ$, т.е. ограничения на геометрию траектории отсутствуют. Условие 1.3 соответствует сложному поведению системы, при котором траектория не покидает секторы *IV*, *I*. При достаточно большом времени моделирования можно сделать заключение о том, что ограничения на геометрию траектории также отсутствуют. Очевидно, что если выполнено условие 1.2 или 1.3, то задано слишком большое значение R_d для выбранных БЛА и условий полета. Отрицательное значение угла **α** означает выбор слишком малого значения R_d.

Для проведения численного моделирования рассмотрим уравнения динамики БЛА. Очевидно, что на квадрокоптер действуют сила тяги *F*, сила лобового сопротивления *X*, а также сила тяжести *G*. После проекции этих сил на полускоростную систему координат имеем [25]:

$$\begin{aligned}
&m\frac{\mathrm{d}V(t)}{\mathrm{d}t}=-F\sin\left(\alpha_c(t)\right)\cos\left(\beta_c(t)\right)-\\
&-X\left(V(t),\alpha_c(t)\right)-G\sin\left(\Theta(t)\right),\\
&mV(t)\frac{\mathrm{d}\Theta(t)}{\mathrm{d}t}=\\
&=F\left(\cos\left(\gamma_c(t)\right)\cos\left(\alpha_c(t)\right)-\sin\left(\gamma_c(t)\right)\sin\left(\alpha_c(t)\right)\sin\left(\beta_c(t)\right)\right)-\\
&-G\cos\left(\Theta(t)\right),\\
&-mV(t)\cos\left(\Theta(t)\right)\frac{\mathrm{d}\Psi(t)}{\mathrm{d}t}=\\
&=F\left(\sin\left(\gamma_c(t)\right)\cos\left(\alpha_c(t)\right)+\cos\left(\gamma_c(t)\right)\sin\left(\alpha_c(t)\right)\sin\left(\beta_c(t)\right)\right),
\end{aligned}\qquad(1)$$

где α_c и β_c – углы атаки и скольжения БЛА соответственно, Θ и Ψ – углы наклона и поворота траектории соответственно.

Поскольку мы рассматриваем сбалансированный полет, то уравнения изменения вращательного движения не потребуются.

Уравнения движения центра тяжести БЛА в $Px_gy_gz_g$ описываются так:

$$\begin{aligned}
\frac{dx_g}{dt}&=V_g\cos(\Theta)\cos(\Psi),\\
\frac{dy_g}{dt}&=V_g\sin(\Theta),\\
\frac{dz_g}{dt}&=-V_g\cos(\Theta)\sin(\Psi).
\end{aligned}\qquad(2)$$

Как видно, три уравнения (2) связывают 8 переменных: $m,V,F,\alpha_c,\beta_c,\gamma_c,\Theta,\Psi$. Теперь, если согласно допущениям выше, зафиксировать пять из них, а именно: $m,V,F\equiv F_{\max},\beta_c\equiv 0,\Theta\equiv 0$, то получим замкнутую систему, из которой сможем найти параметры полета α_c и β_c. Следовательно, сможем провести численное моделирование. Сила лобового сопротивления имеет вид

$$X(V,\alpha_c)=\frac{1}{2}C_x\rho V^2S(\alpha_c),\qquad(3)$$

где C_x – коэффициент лобового сопротивления, ρ – плотность воздуха, $S(\alpha_c)=\pi r^2\cos\left(\alpha_c\right)$ – характерная площадь БЛА, определяемая как

Табл.1. Основные характеристики моделирования

Параметр	Обозначение	Значение
Масса БЛА	m	0.468 кг
Радиус БЛА	r	0.27 м
Максимальная тяга БЛА	F_{max}	11 Н
Воздушная скорость БЛА	V	4.5 м/с
Коэффициент лобового сопротивления БЛА	C_x	0.35
Максимальная скорость ветра	W	2.5 м/с
Плотность воздуха (при температуре 20˚C)	ρ	1.2 кг/м^3
Радиус круга C	R_d	2.69 м
Угол отклонения V_g в точке P	α_V	45˚

площадь эллипса, полученного путем наклона окружности радиуса r на угол α_c.

В Табл.1 представлены основные характеристики, используемые для расчета **α** [24-27]. Радиус R_d выбирался равным размеру одной клетки (см. ниже).

С помощью предложенного алгоритма и численного решения системы (1)-(3) с параметрами Табл.1 имеем: α_c=-4,77˚, β_c=-65,81˚, **α**=24,4˚.

2.2. Метод планирования траектории на плоскости, учитывающий ограничения на форму траектории

Будем рассматривать задачу планирования как задачу поиска пути на графе специального вида – графе регулярной декомпозиции ГРД (в англоязычной терминологии – grid). Формально ГРД - это тройка:

$$\mathbf{Gr}=<A, ADJ, d>,$$

где A – множество клеток, представляющее собой матрицу $A_{m\times n}=\{a_{ij}\}$: $a_{ij}=0 \vee a_{ij}=1$, $\forall i, j$: $0\le i<m$, $0\le j<n$, $m,n\in\mathbf{N}$; $ADJ\subseteq A\times A$ – отношение, задающее смежность на множестве клеток (множество смежных клеток); d – метрика на множестве A (функция, задающая способ определения расстояния между клетками).

Здесь и далее будем периодически использовать запись $a(i; j)$ или просто $(i; j)$ для обозначения клетки a_{ij}. В тех случаях, когда индексные координаты клеток не важны, будем обозначать их буквами латинского алфавита: a, b, c и т.д. Две различные клетки ГРД $(i_1; j_1)$ и $(i_2; j_2)$ будем считать смежными, если $|i_1-i_2|\le 1 \vee |j_1-j_2|\le 1$. Фактически, смежным клеткам соответствуют смежные элементы матрицы $A_{m\times n}$.

Рис. 2. Путь как последовательность клеток ГРД

Будем считать, что в качестве метрики на ГРД используется функция (евклидова метрика):

$$d(a_{ij}, a_{lk})=\sqrt{(l-i)^2+(k-j)^2}\ .$$

Клетку ГРД будем называть проходимой, если $a_{ij}=0$, непроходимой, если $a_{ij}=1$. Множество клеток, смежных с a_{ij} будем обозначать как $adj(a_{ij})$ или как $adj(a(i; j))$. Множество попарно смежных непроходимых клеток будем называть препятствием: $Obs=\{a(i_0; j_0), a(i_1; j_1), \ldots, a(i_s; j_s) \mid a(i_k; j_k)=1, a(i_k; j_k)\in adj(a(i_{k-1}; j_{k-1}))\ \forall k=0,1,\ldots, s, s\in\mathbf{N}\}$.

С использованием ГРД задача планирования траектории **PTask** может быть сформулирована следующим образом. Пусть на ГРД зафиксированы различные проходимые клетки (Рис.2): начальная $s(i_s; j_s)$ и целевая $g(i_g; j_g)$. Необходимо найти путь $\pi(s, g)$, т.е. последовательность клеток $\pi(s, g)=\{a(i_0; j_0), a(i_1; j_1), \ldots, a(i_v; j_v)\}$, такую что $a(i_0; j_0)=s(i_s; j_s)$, $a(i_v; j_v)=g(i_g; j_g)$, $\forall k: 1\le k<v$ $a(i_k; j_k)=0$, $a(i_k; j_k)\in adj(a(i_k-1; j_k-1))$.

Очевидно, что для решения сформулированной в разделе 1 задачи, необходимо наложение определенных ограничений на путь π. Об этом будем сказано ниже. Пока же опишем алгоритм построения ГРД для поставленной задачи, т.е. алгоритм построения графа для заданной на плоскости области U.

Построение ГРД для прямоугольной рабочей области U: $x_{\min}\le x\le x_{\max}$, $y_{\min}\le y\le y_{\max}$ ($U=U_{free}+U_{obs}$) производится по следующему алгоритму:

Шаг 1. Положить $i = 0$, $j = 0$, $p_x=x_{min}+l$, $p_y=y_{max}-l$.

Шаг 2. Если область $[p_x-l, p_x+l]\times[p_y-l, p_y+l]$ содержит хотя бы одну непроходимую точку из U, то добавить в A клетку $a_{ij}=1$, в противном случае – клетку $a_{ij}=0$.

Шаг 3. Положить $j = j + 1$; $p_x = p_x + 2l$.

Шаг 4. Если точка (p_x, p_y) лежит в рабочей области, то перейти к Шагу 2.

Шаг 5. Если $p_x > x_{max}$ и $p_y < y_{min}$, то завершить работу алгоритма.

Шаг 7. Положить $j = 0$; $i = i + 1$; $p_y = p_y - 2l$; $p_x = x_{min}+l$ и перейти к Шагу 2.

Фактически алгоритм осуществляет перебор точек рабочей области, отстоящих друг от друга на расстоянии $2l$, «слева направо, снизу вверх». Для каждой из точек проверяется, содержит ли квадрат со стороной $2l$ и с центром в точке (p_x, p_y) хоть одну непроходимую точку. Если да, то в ГРД добавляется непроходимая клетка, в противном случае – проходимая (Рис.3,в).

Очевидно, что для решения практических задач необходимо, чтобы выполнялось неравенство $l \geq \delta r$, где r – радиус круга, представляющего собой геометрическую модель мультикоптера (см. выше), а $\delta \geq 1$ – некоторый поправочный коэффициент, необходимый, чтобы мультикоптер мог целиком уместиться в одной клетке (Рис.3,б). Однако возможна ситуация, когда геометрический центр мультикоптера находится в некоторой проходимой клетке, а часть мультикоптера - уже в другой, соседней клетке, которая не обязательна будет проходимой. Таким образом, для того, чтобы мог быть осуществлен полет по всем проходимым клеткам, необходимо, чтобы их соседние клетки также были проходимыми. Чтобы это гарантировать сделаем следующее. После построения ГРД по вышеприведенному алгоритму для каждой непроходимой клетки пометим все смежные с ней клетки особым образом и затем положим их непроходимыми (Рис.3,г). Таким образом, мы искусственно расширяем границы непроходимых областей пространства (препятствий) на единичный размер клетки. Эта дополнительная область в реальности является проходимой и создает необходимый запас для того, чтобы осуществить полет по соседним проходимым клеткам даже в том случае, когда часть мультикоптера оказывается за пределами этих клеток. В этом случае каждая псевдонепроходимая клетка с линейным размером $2\delta r$ определяет некоторую окрестность желаемой траектории, которую мы назвали «трубкой».

Задача планирования с ограничениями. Как было сказано выше, предлагается искать решение задачи планирования с одной стороны в виде последовательности отрезков прямых, лежащих в U_{free}, таких что модуль угла, образуемого между любыми смежными отрезками, не превышает порогового значения $\boldsymbol{\alpha}$, с другой стороны – в виде пути $\boldsymbol{\pi}$ на графе регулярной декомпозиции, построенном для рабочей области U по описанному выше алгоритму. Для того, чтобы формализовать ограничения, накладываемые на форму пути ГРД, введем следующие определения.

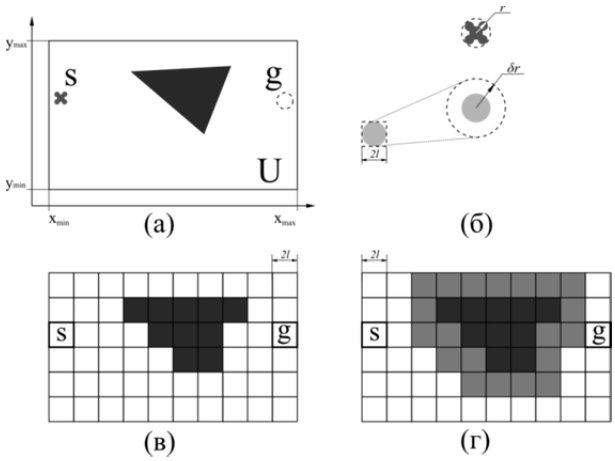

Рис. 3. Рабочая область (а), объект управления (б) и соответствующий ГРД (в-г)

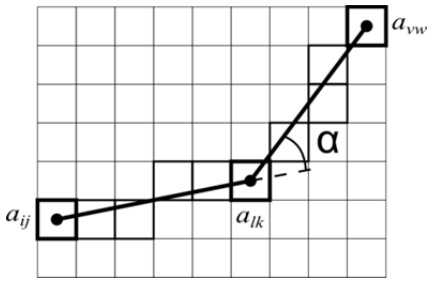

Рис. 4. Угол между двумя смежными секциями

Нуль-траекторией $ntr(a_{ij}, a_{lk})$ между двумя проходимым клетками a_{ij}, a_{lk} будем называть последовательность клеток, построенную по алгоритму Брезенхема [28] – одному из базовых и наиболее распространенных в машинной графике алгоритмов построения дискретных прямых. Нуль-траектории на ГРД соответствует отрезок прямой в рабочей области U, соединяющий центры соответствующих клеток (Рис. 4).

Упорядоченную пару клеток ГРД будем называть секцией и обозначать как $e=\langle a_{ij}, a_{lk}\rangle$. Весом секции будем называть величину $c(e)=c\langle a_{ij}, a_{lk}\rangle=d(a_{ij}, a_{lk})$. Будем считать секцию $e=\langle a_{ij}, a_{lk}\rangle$ проходимой, если соответствующая нуль-траектория $ntr(a_{ij}, a_{lk})$ содержит лишь проходимые клетки. Две секции, имеющие хотя бы

одну общую клетку $e_1=\langle a_{ij}, a_{lk}\rangle$, $e_2=\langle a_{lk}, a_{vw}\rangle$, будем называть смежными. Углом отклонения секции e_2 от e_1 будем называть модуль угла между векторами $\overrightarrow{a_{ij}a_{lk}}$ и $\overrightarrow{a_{lk}a_{vw}}$, координаты которых равны $(l - i, k - j)$ и $(v - l, w - k)$ соответственно.

Частичным путем из начальной клетки в целевую будем называть последовательность клеток ГРД $PP(s, g)=\{a(i_0; j_0), a(i_1; j_1), \ldots, a(i_v; j_v)\}$ такую, что $a(i_0; j_0)=s$, $a(i_v; j_v)=g$. Клетки, входящие в частичный путь будем называть опорными. Аналогично можно определить частичный путь как последовательность смежных секций $e_1, e_2, \ldots, e_v$. Величину $\alpha_m(PP)=\alpha_m=\max\{\alpha(e_1, e_2), \alpha(e_2, e_3), \ldots, \alpha(e_{v-1}, e_{v-1})\}$ будем называть максимальным углом отклонения частичного пути PP. Весом частичного пути будем называть сумму весов образующих его секций: $C(PP)=c(e_1)+c(e_2)+\ldots+c(e_v)$. Вес частичного пути на ГРД соответствует длине соответствующей траектории в рабочей области U.

Теперь задача планирования траектории может быть сформулирована как задача поиска частичного пути, удовлетворяющего следующим ограничениям: все секции частичного пути проходимы; максимальный угол отклонения частичного пути не превышает заранее заданного значения $\boldsymbol{\alpha}$.

Частичный путь, удовлетворяющий этим условиям, будем называть допустимым. Очевидно, что в общем случае может существовать множество различных допустимых частичных путей, являющихся решением задачи планирования. Рассмотрим множество всех возможных допустимых частичных путей из s в g: $\boldsymbol{PP}=\{PP_1, PP_2,\ldots, PP_n\}$. Назовем кратчайшим такой частичный путь PP_i, что $\forall j\neq i\ c(PP_j)\geq c(PP_i)$, т.е. кратчайший путь – это путь с минимальным весом. Кратчайший путь будем считать оптимальным решением задачи планирования траектории.

Алгоритм планирования. К настоящему моменту авторам работы не известны методы поиска пути на ГРД, учитывающие ограничения на геометрическую форму этого пути – в частности ограничения на максимальный угол отклонения. Известны методы, в которых, так или иначе, вводятся понятия, схожие с приведенными выше понятиями нуль-траектории (line of sight), секции, угла между секциями и т.д. Однако они используются в основном для модификации известных методов с целью повышения качества решения задачи планирования (снижения веса отыскиваемых путей), а также – вычислительной эффективности [29, 30].

Предлагаемый нами алгоритм поиска пути на ГРД, учитывающий ограничения на максимальный угол отклонения LIAN (limited angle), может рассматриваться как модификация алгоритма A* [12]. Другим близким алгоритмом к LIAN является R* [15]. Так же как и A*, предлагаемый алгоритм осуществляет обход графа и расчет g-значений вершин (клеток) $g(a)$, где g-значение – это вес кратчайшего пути в клетку, известный к текущей итерации алгоритма: $g(a)=c(\pi^*(s, a))$. Помимо g-значения каждая клетка обязательно характеризуется родительским указателем (в отличие от алгоритма A*, где хранение родительских указателей является опциональным) – указателем на клетку, предшествующую данной: $bp(a)$. При этом у клетки с одними и теми же координатами – в отличие от A* – может быть несколько родителей (этим алгоритм схож с R*): $bp_1(a)\neq bp_2(a)\neq\ldots\neq bp_K(a)$. Таким образом, когда речь идет о клетке LIAN имеется в виду тройка [клетка, g-значение, родительский указатель] ($[a, g(a), bp(a)]$).

LIAN, как и другие алгоритмы эвристического поиска семейства A*, поддерживает в упорядоченном состоянии список *OPEN*: список клеток (в описанном выше смысле) – потенциальных кандидатов на дальнейшее рассмотрение (изначально содержащий лишь клетку $[s, 0, \varnothing]$). На каждом шаге алгоритма из списка *OPEN* выбирается клетка с минимальным f-значением $f(a)$, где $f(a)=g(a)+h(a)$, $h(a)$ – эвристическая оценка веса пути из клетки a в целевую клетку, $h(a)=d(a, g)$. Т.е. выбирается клетка, потенциально лежащая на кратчайшем пути из s в g (аналогично A*).

Затем строится и проверяется на проходимость нуль-траектория $ntr(bp(a), a)$. Если она непроходима, то клетка $a=[a, g(a), bp(a)]$ исключается из дальнейшего рассмотрения, если нуль-траектория проходима то для клетки a, генерируются потенциальные последователи (вершины, для которых клетка a – родительская): $SUCC(a)=\{succ_1(a), succ_2(a), \ldots, succ_N(a) \mid a=bp(succ_i(a))\ \forall i:1..N\}$. В отличие от A*, где потенциальными последователями считаются все клетки, смежные с данной. Здесь к потенциальным последователям относятся клетки, расположенные на фиксированном расстоянии $\Delta\in\boldsymbol{N}$ от a. Для идентификации таких клеток строится

дискретная окружность с центром в клетке a радиусом Δ по известному в машинной графике алгоритму Midpoint [31]– Рис. 5. При этом, если расстояние до целевой клетки меньше радиуса окружности, то целевая клетка также добавляется в список *SUCC*.

Далее из списка *SUCC* удаляются те клетки, которые не удовлетворяют заданному ограничению на угол отклонения, т.е. клетки $succ_i(a)$: $\alpha(\langle bp(a), a\rangle, \langle a, succ(a)\rangle)>\boldsymbol{\alpha}$. (при этом, если родителем клетки является начальная клетка, ограничения на угол отклонения игнорируются). Также из списка *SUCC* удаляются клетки, рассмотренные ранее. У оставшихся клеток фиксируется g-значение, $g(succ_i(a))=g(a)+d(a, succ_i(a))$, и они заносятся в список *OPEN*. Сама клетка a заносится в список рассмотренных клеток *CLOSED*. Алгоритм продолжает работу до тех пор, пока в список *CLOSED* не будет добавлена целевая клетка (Рис.6). После того как это произойдет искомый путь может быть восстановлен с использованием родительских указателей $PP=\{\ldots, bp(bp(g)), bp(g), g\}$.

Предложенный алгоритм обладает следующими свойствами.

Свойство 1. Алгоритм LIAN корректен, т.е. он всегда завершает свою работу.

Идея доказательства. Алгоритм осуществляет свою работу, пока список *OPEN* не пуст (либо пока не будет найден искомый допустимый частичный путь). Очевидно, что в список *OPEN* могут добавляться элементы, соответствующие клеткам ГРД, число которых конечно. Число потенциальных родителей у каждой клетки также конечно. При этом на этапе добавления элемента в список *OPEN* проверяется – не был ли этот элемент (клетка с совпадающими индексными координатами и родительским указателем) рассмотрен ранее. Таким образом, число элементов в списке *OPEN* ограничено сверху. Поскольку на каждом шаге алгоритма всегда удаляется один элемент из списка *OPEN* (строка 8), рано или поздно этот список будет исчерпан, либо будет найден путь. В обоих случаях (строки 6 и 12) алгоритм завершает свою работу.

Свойство 2. Алгоритм LIAN полон (для фиксированного Δ), т.е. он всегда находит искомый путь, либо возвращает failure, что свидетельствует о том, что пути (при заданном Δ) не существует.

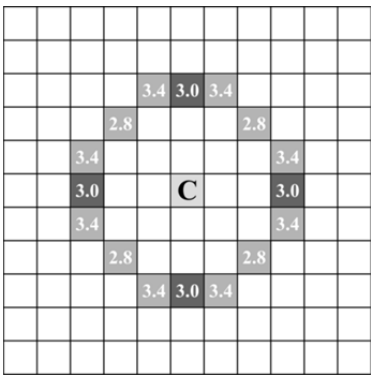

Рис. 5. Окружность радиусом в 4 клетки

Числа внутри клеток равны округленному расстоянию до центра

Идея доказательства. Значение Δ однозначно определяет множество возможных последователей для каждой рассматриваемой клетки a, при этом все клетки из этого множества, удовлетворяющие ограничению на максимальный угол отклонения, добавляется в список *OPEN*. Т.е. рано или поздно будут рассмотрены все возможные допустимые частичные пути, состоящие из секций длины Δ, и искомый путь (если он существует) будет найден.

Свойство 3. Алгоритм LIAN оптимален (для фиксированного Δ), т.е. он всегда находит кратчайший допустимый частичный путь (если он существует).

Идея доказательства. Алгоритм LIAN использует такую же эвристическую стратегию поиска в пространстве состояний, что и алгоритм A*, который гарантирует отыскание кратчайшего пути при условии использования монотонной эвристики. Поскольку используемая LIAN в качестве эвристики функция d удовлетворяет определению монотонности, то LIAN (так же как и A*) гарантирует отыскание кратчайшего пути (если такой существует при заданный ограничениях на максимальный угол отклонения и при фиксированном Δ).

Заметим также, что поскольку предлагаемый алгоритм использует для приоритезации списка *OPEN* функцию $f(a)=g(a)+h(a)$ аналогичную используемой в A*, то также как и A*, алгоритм LIAN допускает взвешивание эвристики. Т.е. использование в качестве эвристики функции $f(a)=g(a)+w\cdot h(a)$, где $w\geq 1$ – настроечный коэффициент (вес эвристики), влияние которого на эффективность алгоритма A* изучено достаточно хорошо: чем больше вес, тем быстрее алгоритм завершает свою работу. Однако ис-

```
GetPathFromBP()
a = g                                                          (1)
while (a≠s)                                                    (3)
    path = a + path                                            (4)
    a=bp(a)                                                    (5)
return s + path                                                (6)

GenerateSuccessors(a,Δ,α,CLOSED)
SUCC={}                                                        (1)
Построить дискретную окружность радиуса Δ с центром в
клетке a с помощью алгоритма MidPoint - CIRCLE                 (2)
foreach a'∈CIRCLE                                              (3)
    if a'=1 then continue                                      (5)
    if (a ≠ s) and (α(⟨bp(a), a⟩, ⟨a, a'⟩) > α) then            (6)
        continue                                               (7)
    if a'∈CLOSED then continue                                 (9)
    SUCC=SUCC ∪ {a'}                                           (10)
if d(a, goal)<Δ and α(⟨bp(a), a⟩, ⟨a, goal⟩) ≤ α then          (11)
    SUCC=SUCC ∪ {goal}                                         (12)
return SUCC

LIAN(Δ,α)
g(start)=0                                                     (1)
bp(start)=start                                                (2)
g(a)= +∞, ∀a ≠ start                                           (3)
OPEN={start}                                                   (4)
CLOSED={}                                                      (5)

while (OPEN≠{})                                                (6)
    a=argmin_{a∈OPEN} f(a)                                     (7)
    OPEN=OPEN\{a}                                              (8)
    if ntr(bp(a), a) проходима then                            (9)
        CLOSED=CLOSED ∪ {a}                                    (10)
        if a=g then                                            (11)
            return GetPathFromBP()                             (12)

        SUCC=GenerateSuccesors(a,Δ,α)                          (13)
        foreach succ(a)∈SUCC                                   (14)
            g(succ(a))=g(a) + d(a, succ(a))                    (15)
            bp(succ(a))=a                                      (16)
            OPEN=OPEN ∪ {a}                                    (17)

return failure                                                 (18)
```

Рис. 6. Алгоритм LIAN

пользование веса $w>1$ в общем случае не гарантирует отыскание кратчайшего пути (при этом, как показывает практика, отыскиваемые пути отличаются по весу от кратчайших незначительно).

Предложенный алгоритм может быть легко модифицирован для учета ограничений, описанных в разделе 2.1, а именно ограничений предполагающих отсутствие препятствий (непроходимых клеток ГРД) в заданной окрестности точек излома траектории (начальной и конечной клеток каждой из секций, образующих допустимый частичный путь). Для этого на каждой итерации алгоритма, при рассмотрении проходимости нуль-траектории $ntr(bp(a), a)$ – строка 9 – необходимо проверять также, чтобы все клетки смежные с a были проходимы. Здесь и далее будем предполагать, что именно такой модифицированный алгоритм LIAN используется для решения задачи планирования.

3.Экспериментальный анализ

В рамках исследования был проведен экспериментальный анализ работы алгоритма при решении задач планирования траектории маловысотного полета малого (до 1 м. в диаметре) мультикоптера в городских условиях. Для проведения экспериментов было использовано 205 заданий поиска пути на ГРД, характеризующихся следующими особенностями:

1) для построения ГРД использовалась карта Москвы, при этом каждому ГРД соответствовал фрагмент карты размером 1347 x 1347 метров (геоинформационные данные были взяты из открытой цифровой модели местности OSM [23]);

2) всего был использован 81 различный фрагмент карты (81 фрагмент * 5 различных положений начальной и целевой клеток = 205 заданий);

3) размер отдельного ГРД составляет 501 x 501 клеток, т.е. одна клетка соответствует области ~2,7 x 2,7м;

4) плотность препятствий (количество непроходимых клеток по отношению к общему количеству клеток) варьируется от 0,25 до 0,35;

5) положение начальной и целевой клеток выбирается случайно, но так, чтобы расстояние между ними составляло не менее 500 (клеток).

Исследование проводилось на ЭВМ следующей характеристики: AMD Athlon 64 processor 2800+ 1.8GHz, 1.25 ГБ ОЗУ, 64-разрядная операционная система Windows 7.

Перед проведением экспериментов было проведено предварительное тестирование алгоритма LIAN, направленное на выявление зависимости производительности алгоритма от настроечных параметров: Δ - расстояния от текущей рассматриваемой клетки до потенциальных последователей и w - веса эвристики. Ограничения на максимальный угол отклонения составляли 20, 30, 40 либо 180°. Последнее условие фактически означает, что ограничения на угол отклонения отсутствуют.

В ходе предварительных экспериментов было выявлено, что время обработки отдельных заданий превышало один час. Поэтому с целью обеспечения разумных сроков проведения экспериментальных исследований было установлено ограничение на количество итераций цикла поиска пути: 10 000 итераций (что соответствует примерно 5 минутам работы на ЭВМ указанной выше характеристики). Если алгоритм LIAN достигает ограничения на количество итераций, его выполнение прекращается (при этом считается, что путь не найден).

По результатам предварительной серии экспериментов было установлено, что на практике наиболее целесообразно (с точки зрения эффективного использования вычислительных ресурсов) применение алгоритма LIAN с параметрами Δ=10 и w=2.

Основная серия экспериментов заключалась в сравнении алгоритма LIAN со следующими известными алгоритмами поиска: A* [12], WA* (алгоритм A*, использующий взвешенную эвристику), R* [15]. Заметим, что алгоритмы WA* и R*, так же как и LIAN, являются параметризуемыми. Для установления наилучшего значения параметра алгоритма WA* (веса эвристики) была проведена дополнительная серия экспериментов, по результатам которой было выбрано значение w=3. Значения параметров алгоритма R* выбирались согласно эвристическому правилу, предложенному в [32].

Для сравнения результатов работы алгоритмов использовались следующие выходные параметры (индикаторы):

1) Nodes – максимальное количество клеток, информация о которых хранится в оперативной памяти во времена работы алгоритма (характеризует емкостную сложность алгоритма);

2) Length – длина найденного алгоритмом пути (обратим внимание, что при подсчете длины пути, найденного алгоритмом LIAN, учитываются длины секций, в то время как при аналогичном подсчете для алгоритмов A*, WA* и R* учитываются длины переходов между всеми смежными клетками, образующими искомый путь);

3) Time – время работы алгоритма (в секундах);

4) Suc. maps – число ГРД, на которых алгоритм завершился успешно, т.е. путь был найден.

Усредненные значения отслеживаемых индикаторов, полученные в ходе эксперимента, приведены в Табл.2.

Отдельно следует сказать о количестве успешно завершенных заданий (Suc. maps). Так как начальная и целевая клетки выбирались случайным образом, для некоторых заданий решений не существует в принципе (а именно для двух заданий, поэтому известные алгорит-

Табл.2. Результаты экспериментов

	Asearch	WAsearch	Rsearch	liansearch angle = 20	liansearch angle = 30	liansearch angle = 40	liansearch angle = 180
Nodes	15466	2274	510	668	823	1092	2993
Length	576,69	621,58	625,16	580,47	577,64	577,04	579,16
Time	5,231273	0,035369	0,041237	0,023591	0,031838	0,058355	0,14481
Suc. maps	203	203	203	126	151	154	199

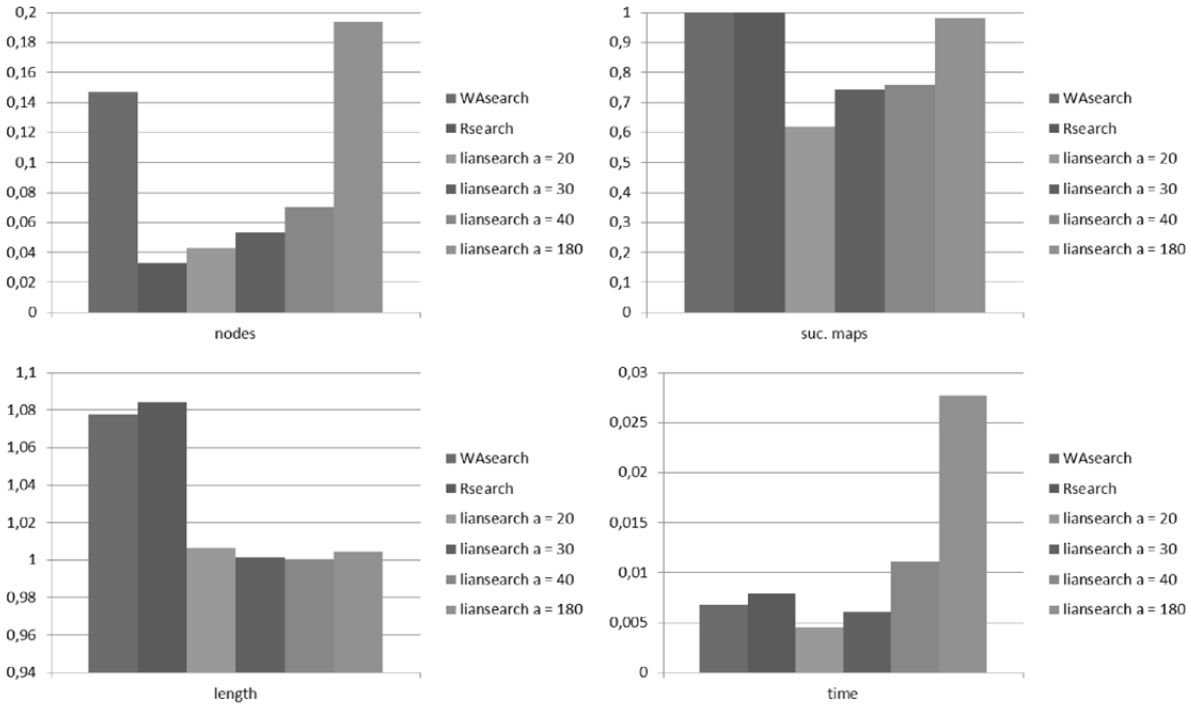

Рис. 7. Нормированные результаты экспериментов

мы успешно завершились только в 203 случаях из 205). Среди заданий, которые не были успешно обработаны алгоритмом LIAN, есть как задания, решений которых не существует при заданных ограничениях на угол отклонения, так и задания, при о бработке которых был превышен лимит числа итераций. Нормированные (относительно показателей A*) результаты можно наблюдать на Рис. 7.

Анализируя представленные результаты, можно утверждать, что результативность и вычислительная эффективность алгоритма LIAN существенно зависит от заданного угла отклонения. Чем строже ограничения, тем меньше вероятность отыскания решения (за ограниченное число итераций), и тем больше времени (и памяти) требуется алгоритму. И даже в том случае, когда ограничения на максимальный угол отклонения отсутствуют, необходимость фиксации расстояния от текущей рассматриваемой клетки до потенциальных последователей (входной параметр Δ) препятствует нахождению допустимого решения задачи планирования (хотя, возможно, при других значениях Δ путь был бы найден). Таким образом, весьма перспективным является проведение дальнейших исследований и модификаций предложенного алгоритма, направленных на устранение этого недостатка (например, путем динамического изменения Δ (некоторым способом) в ходе работы алгоритма).

В завершение заметим, что сравнение с алгоритмами A*, WA* и R* лишь повышает наглядность результатов экспериментальных исследований, но не отрицает того факта, что указанные алгоритмы поиска пути на ГРД не

учитывают ограничения на динамику полета БЛА и допустимое управление, и, следовательно, не могут напрямую применяться для решения поставленной задачи.

Заключение

В работе описан новый подход к решению задачи планирования траектории беспилотного летательного аппарата мультикоптерного типа в условиях ограничений на динамику полета. Предложен оригинальный метод определения ограничений на геометрию полета по имеющейся модели движения и новый метод планирования, учитывающий эти ограничения. Применение описанного подхода позволяет добиться следующих преимуществ по сравнению с существующими аналогами: на этапе планирования не требуется выполнять ресурсоемкое моделирование полета; построенный планировщиком путь реализуем при некоторых допустимых условиях (заданных режиме полета, ветровых нагрузках и т.д.), т.е. учитывает ограничения на динамику движения объекта управления, задаваемые математической моделью.

Результаты проведенных экспериментов показали, что предложенный метод планирования может быть использован для решения задачи построения траектории полета мультикоптера в городских условиях. Тем не менее, следует подчеркнуть, что не каждая поставленная задача может иметь решение с учетом ограничений. Кроме того, в некоторых частных случаях решение задачи может потребовать существенных вычислительных ресурсов, т.е. предметом дальнейших исследований должно явиться совершенствование предложенного алгоритма и разработка новых алгоритмов планирования с учетом ограничений.

Литература

1. Осипов Г.С., Тихомиров И.А., Хачумов В.М., Яковлев К.С. Интеллектуальное управление транспортными средствами: стандарты, проекты, реализации // Авиакосмическое приборостроение. 2009. № 6. с. 34–43.
2. Яковлев К.С., Макаров Д.А., Панов А.И., Зубарев Д.В. Принципы построения многоуровневых архитектур систем управления беспилотными летательными аппаратами // Авиакосмическое приборостроение. 2013. № 4. с. 10-28.
3. Kendoul F. Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems // Journal of Field Robotics. 2012. V. 29. №. 2. P. 315-378.
4. Яковлев К.С., Петров А.В., Хитьков В.В. Программный комплекс навигации и управления беспилотными транспортными средствами // Информационные технологии и вычислительные системы. 2013. № 3. С. 72-83.
5. Yap P. Grid-based path-finding // Advances in Artificial Intelligence. Berlin Heidelberg: Springer, 2002. P. 44-55.
6. Яковлев К. С. Графы специальной структуры в задачах планирования траектории // Труды III международной конференции «Системный анализ и информационные технологии САИТ-2009». 2009.
7. Wooden D. T. Graph-based path planning for mobile robots : дис. Georgia Institute of Technology. 2006.
8. Kavraki L. E. et al. Probabilistic roadmaps for path planning in high-dimensional configuration spaces // Robotics and Automation, IEEE Transactions on. 1996. V. 12. № 4. P. 566-580.
9. Яковлев К.С., Баскин Е.С. Графовые модели в задаче планирования траектории на плоскости // Искусственный интеллект и принятие решений. 2013. №1. С. 5-12.
10. Dijkstra E.W. A note on two problems in connexion with graphs //Numerische mathematik. 1959. V.1. №. 1. P. 269-271.
11. Pearl J. Heuristics. Reading, Massachusetts: Addison-Wesley Publishing Company, 1984.
12. Hart P. E., Nilsson N. J., Raphael B. A formal basis for the heuristic determination of minimum cost paths // Systems Science and Cybernetics, IEEE Transactions on. 1968. V.4. №. 2. P. 100-107.
13. Korf R.E. Artificial intelligence search algorithms // CRC Handbook of Algorithms and Theory of Computation, M.J. Atallah (Ed.). Boca Raton, FL: CRC Press, 1998.
14. Variants of A* // [Электронный ресурс] URL: http://theory.stanford.edu/~amitp/GameProgramming/Variations.html (дата обращения: 20.08.2014).
15. Likhachev M., Stentz A. R* Search. // In Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence. Menlo Park, Calif. 2008
16. Яковлев К.С. HGA*: эффективный алгоритм планирования траектории на плоскости // Искусственный интеллект и принятие решений. 2010. № 2. С. 16-25.
17. Latombe J. Robot Motion Planning. Boston: Kluwer Academic Publishers, 1991.
18. Kim H., Park B., Myung H. Curvature Path Planning with High Resolution Graph for Unmanned Surface Vehicle // Robot Intelligence Technology and Applications. 2013. P. 147-154.
19. LaValle S. M., Kuffner Jr J. J. Rapidly-exploring random trees: Progress and prospects. In Algorithmic and Computational Robotics: New Directions. 2000.
20. Bertram D. et al. An integrated approach to inverse kinematics and path planning for redundant manipulators // Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on. 2006. P. 1874-1879.
21. Жулев В.И., Леушкин В.С., Нгуен Т.Н. Планирование локальной траектории автомобиля-робота в реальном времени // Вестник РГРТУ. 2013. № 4 (46). Часть 3.
22. Корытов М.С. Автоматизация синтеза оптимальных траекторий перемещения грузов мобильными грузоподъемными кранами в неоднородном организованном трехмерном пространстве. Омск: СибАДИ, 2012. 380 с.
23. Haklay M., Weber P. Openstreetmap: User-generated street maps // Pervasive Computing. 2008. V. 7. №. 4. P. 12-18.

24. AscTec Hummingbird // [электронный ресурс] URL: http://www.asctec.de/en/uav-uas-drone-products/asctec-hummingbird/ (дата обращения: 20.08.2014)
25. Лебедев А.А., Чернобровкин Л. С. Динамика полета. М.: Оборонгиз, 1962
26. Schoellig A.P., Mueller F.L., D'Andrea R. Optimization-based iterative learning for precise quadrocopter trajectory tracking // Autonomous Robots. 2012. V.33. №.1-2. P.103-127.
27. Brescianini D., Hehn M., D'Andrea R. Quadrocopter pole acrobatics // Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference. 2013. P. 3472-3479.
28. Bresenham J. E. Algorithm for computer control of a digital plotter //IBM Systems journal. 965. V. 4. №. 1. P. 25-30.
29. Daniel K. et al. Theta*: Any-angle path planning on grids //Journal of Artificial Intelligence Research. 2010. V. 39. №. 1. P. 533-579.
30. Munoz P., Rodriguez-Moreno M. Improving efficiency in any-angle path-planning algorithms // Intelligent Systems (IS), 2012 6th IEEE International Conference. 2012. P. 213-218.
31. Pitteway M. L. V. Algorithms of conic generation // Fundamental algorithms for computer graphics. Springer Berlin Heidelberg, 1991. – C. 219-237.
32. 32. Yakovlev, K., Baskin, E., & Hramoin, I. (2014). Fine-tuning Randomized Heuristic Search for 2D Path Planning: Finding the Best Input Parameters for R* Algorithm through Series of Experiments. In Artificial Intelligence: Methodology, Systems, and Applications (pp. 278-285). Springer International Publishing.

Яковлев Константин Сергеевич. Старший научный сотрудник лаборатории интеллектуальных динамических систем ИСА РАН. Кандидат физико-математических наук. Автор 20 печатных работ. Область научных интересов: искусственный интеллект, робототехника, когнитивные науки, интеллектуальные динамические системы, интеллектуальные системы управления, планирование, планирование траектории, моделирование целенаправленного поведения, эвристический поиск. E-mail: yakovlev@isa.ru

Макаров Дмитрий Александрович. Научный сотрудник ИСА РАН. Окончил Рыбинскую государственную авиационную технологическую академию им. П.А. Соловьева в 2008 году. Кандидат физико-математических наук. Автор 15 печатных работ. Область научных интересов: управление сложными динамическими системами, робастные методы устойчивости и стабилизируемости, искусственный интеллект, экспертные системы. E-mail: makarov@isa.ru

Баскин Егор Сергеевич. Аспирант ИСА РАН. Окончил Московский государственный технический университет им. Баумана. Автор трех печатных работ. Область научных интересов: искусственный интеллект, интеллектуальные динамические системы, автоматическое планирование, теория автоматического управления. E-mail: baskin@isa.ru