УДК 681.786.23 +629.735-519 *DOI: 10.18287/2541-7533-2021-20-3-37-48*

СКАНИРУЮЩИЙ ЛАЗЕРНЫЙ ДАЛЬНОМЕР НА ПОДВЕСЕ С ДВУМЯ СТЕПЕНЯМИ СВОБОДЫ

В. А. Зеленский доктор технических наук, доцент, профессор кафедры конструирования и технологии электронных систем и устройств;
Самарский национальный исследовательский университет имени академика С.П. Королёва;
vaz-3@yandex.ru

М. В. Капалин аспирант кафедры конструирования и технологии электронных систем и устройств;
Самарский национальный исследовательский университет имени академика С.П. Королёва;
no95typem@yandex.ru

Представлен сканирующий лазерный дальномер, закреплённый на подвесе с двумя степенями свободы, который может быть использован в составе навигационной системы беспилотного летательного аппарата для обхода препятствий или предотвращения столкновений. По сравнению со стереокамерами устройство требует значительно меньше вычислительных ресурсов, менее зависимо от условий освещения. По сравнению с интегральными сканирующими лидарами устройство имеет на порядок меньшую стоимость. Разработана модель устройства и выполнена имитация полёта с обходом препятствий в симуляторе «Gazebo». В качестве автопилота использовано программное обеспечение «PX4/Avoidance». В результате модельного эксперимента установлено, что сканирующий лазерный дальномер может обеспечить автономную навигацию с обходом препятствий.

Сканирующий лазерный дальномер; подвес с двумя степенями свободы; беспилотный летательный аппарат; обход препятствий; автопилот PX4/Avoidance; симулятор Gazebo

Цитирование: Зеленский В.А., Капалин М.В. Сканирующий лазерный дальномер на подвесе с двумя степенями свободы // Вестник Самарского университета. Аэрокосмическая техника, технологии и машиностроение. 2021. Т. 20, № 3. С. 37-48. DOI: 10.18287/2541-7533-2021-20-3-37-48

Введение

В настоящее время беспилотные летательные аппараты (БПЛА) используются для решения множества задач. Но их применение в условиях сред с препятствиями остаётся ограниченным. Для полёта в таких средах БПЛА должен обладать интеллектуальной системой управления, которая может обнаруживать препятствия и принимать меры по предотвращению столкновений с ними. Функция «обход препятствий» предполагает планирование безопасного пути, а функция «предотвращение столкновений» обеспечивает непосредственную защиту от столкновений. Первая функция является более сложной, но необходимой для автономной навигации, вторая может применяться как в режиме автоматического, так и ручного управления БПЛА.

Для реализации любой из вышеназванных функций БПЛА должен получать информацию об окружающих его объектах с помощью телекоммуникационных устройств и/или датчиков. Наиболее распространено использование стереокамер. Однако камеры имеют ряд недостатков, среди которых наиболее значительными являются зависимость от условий видимости и требование определённого уровня производительности аппаратного обеспечения на борту летательного аппарата. Многообещающе выглядит использование лидаров. Уже существуют достаточно миниатюрные варианты для использования на БПЛА, такие как лидар Puck компании Velodyne [1] или Leddar Vu 8 Channel Module [2]. Фактором, сдерживающим их широкое применение, является стоимость.

Например, лидар Puck имеет стоимость около 8000 USD, а лидар компании Leddar – около 800 USD [3], в то время как его угол обзора составляет всего 3 градуса по вертикали.

Тем не менее, лазерное излучение позволяет очень точно определять расстояние до точки в пространстве, практически не зависит от освещения и погодных условий. Потому применение этих датчиков в области навигации БПЛА остается актуальным. Для автономной навигации БПЛА с обходом препятствий предлагается использование сканирующего лазерного дальномера (СЛД) на подвесе с двумя степенями свободы.

В работе поставлена задача разработать модель СЛД, закрепленного на подвесе с двумя степенями свободы, а также провести модельный эксперимент полёта БПЛА с этим устройством в режиме автономной навигации с возможностью обхода препятствий.

Структурная схема, математическая модель и алгоритм работы устройства

Принцип работы сканирующего лазерного дальномера показан на рис. 1. В состав СЛД входят мотор для поворота в вертикальной плоскости *2*, рама для крепления устройства к БПЛА *3*, мотор для поворота в горизонтальной плоскости *4*, подвес с двумя степенями свободы *5*, лазерный дальномер *6*. Точка в пространстве, до которой будет измерено расстояние, обозначена цифрой *1*, исходящий и отраженный лазерные лучи отмечены цифрой *7*, траектория движения исходящего луча при сканировании обозначена цифрой *8*.

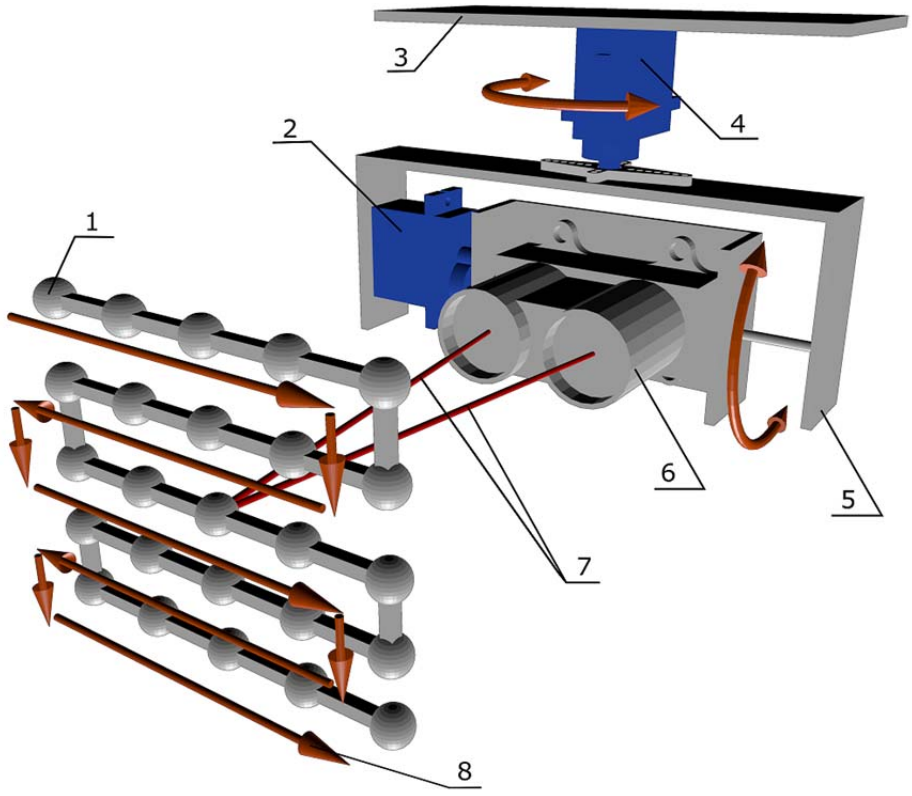

Рис. 1. Принцип работы сканирующего лазерного дальномера на подвесе с двумя степенями свободы

Сканирование происходит в соответствии с алгоритмом:

1. Из краевой точки дальномер будет вращаться сначала по горизонтали до противоположной краевой точки, измеряя данные в $m+1$ точках, где m – количество шагов на горизонтальных этапах.

2. Затем происходит поворот по вертикали на 1 шаг.

3. После этого дальномер снова вращается по горизонтали (в противоположном направлении относительно направления поворота на предыдущем горизонтальном этапе).

4. Затем снова происходит поворот по вертикали на 1 шаг.

5. И так далее, до выполнения всех вертикальных шагов.

6. После этого устройство осуществляет движения в обратном порядке.

Если перед устройством расположена плоскость, то траектория точки, до которой измеряется расстояние, будет соответствовать изображённой на рис. 1. Данная траектория отражает лишь половину цикла, представленного в алгоритме. Вторая половина цикла будет отображаться траекторией, имеющей обратное направление.

Путём вращения моторов можно изменять пространственную ориентацию дальномера, но делать это можно с ограниченной скоростью. Частота измерений самого дальномера ограничивается его характеристиками. Таким образом, максимальное количество точек N, дальность до которых может быть измерена в течение одной секунды, ограничена и определяется следующим выражением:

$$N_{\max} = \min\left[\left(\frac{\upsilon_h}{\Delta_h}\left(\frac{n_h}{n_h+n_v}\right)+\frac{\upsilon_v}{\Delta_v}\left(\frac{n_v}{n_h+n_v}\right)+1\right),\nu\right] ,$$

где υ_h – скорость поворота мотора по горизонтали, град./с; Δ_h – шаг сканирования по горизонтали, град.; n_h – суммарное количество шагов по горизонтали; n_v – суммарное количество шагов по вертикали; υ_v – скорость поворота мотора по вертикали, град./с; Δ_v – шаг сканирования по вертикали, град.; ν – частота измерения дальномера, Гц (кол-во измерений/с).

Сделаем допущение, что скорости и шаги по горизонтали и вертикали одинаковы:

$$\upsilon_h = \upsilon_v = \upsilon;$$
$$\Delta_h = \Delta_v = \Delta .$$

Тогда

$$N_{\max} = \min\left(\frac{\upsilon}{\Delta}+1,\nu\right) ,$$

где υ – скорость поворота моторов, град./с; Δ – шаг сканирования, град.

Если в качестве двигателей будут использованы сервомоторы, то большинство из них имеют максимальную скорость поворота около 500 °/с. Частота измерений дальномера зависит от модели и может достигать значений 100…1000 Гц. В данной работе были использованы характеристики дальномера «LIDAR-Lite 3 Laser Rangefinder High Performance (LLV3HP)» [4] для симуляции. У данного дальномера частота измерений

равна 500 Гц. В таком случае, даже при шаге в один градус, $N_{\max}$ будет ограничиваться выражением

$$N_{\max} = \frac{\upsilon}{\Delta} + 1 \ .$$

На практике могут возникнуть соображения выбора различных шагов по горизонтали и вертикали.

Слишком малые значения шагов либо приведут к недостаточному размеру области обзора, либо низкой частоте сканирования области. Чтобы показать взаимосвязь величины шага, угла обзора и частоты сканирования, рассмотрим следующие выражения.

Пусть n_{h1} – количество шагов по горизонтали на 1 линии сканирования. Учитывая алгоритм сканирования, представленный выше, можно выразить эту величину в виде:

$$n_{h1} = \frac{n_h}{n_v + 1} \ .$$

Определим FOV_h – угол обзора по горизонтали и FOV_v – угол обзора по вертикали в градусах (Field of view, область обзора):

$$FOV_h = \Delta_h n_{h1} = \Delta_h \frac{n_h}{n_v + 1};$$
$$FOV_v = \Delta_v n_v \ .$$

Чем больше эти значения, тем шире зона сканирования и тем больше вероятность, что будет выбран правильный путь обхода препятствия. Уменьшение шага приведёт к сокращению угла обзора при неизменном количестве точек. Если же количество точек увеличивать для сохранения значения угла обзора, то тогда будет уменьшаться количество полных циклов сканирования за одну секунду. Для иллюстрации этого введём параметр α – результирующий угол поворота, на который происходит поворот за 1 цикл сканирования (град.):

$$\alpha = FOV_h \left(\frac{FOV_v}{\Delta_v} + 1 \right) + FOV_v \, .$$

Или, если выразить через шаги сканирования:

$$\alpha = \Delta_h n_{h1} \left(n_v + 1 \right) + \Delta_v n_v \, .$$

Результирующее время перемещения за одно сканирование:

$$t_{scan} = \frac{1}{u_h} FOV_h \left(\frac{FOV_v}{\Delta_v} + 1 \right) + \frac{1}{u_v} FOV_v .$$

Если выразить это время через количество шагов, последнее выражение приобретает вид:

$$t_{scan} = \Delta_h n_{h1} \left(n_v + 1 \right) + \Delta_v n_v .$$

Тогда FPS (Frame Per Second, кадров в секунду) – количество полных циклов сканирований в секунду, или, по-другому, частота формирования полного облака точек от информации с предлагаемого устройства (Гц):

$$FPS = \left(t_{scan} \right)^{-1} = \left(\frac{1}{u_h} FOV_h \left(\frac{FOV_v}{\Delta_v} + 1 \right) + \frac{1}{u_v} FOV_v \right)^{-1},$$

$$FPS = \left(t_{scan} \right)^{-1} = \left(\Delta_h n_{h1} \left(n_v + 1 \right) + \Delta_v n_v \right)^{-1}.$$

Аналогично можно определить количество измерений N за 1 цикл сканирования (ед.):

$$N = \left(\frac{FOV_v}{\Delta_v} + 1 \right) \cdot \left(\frac{FOV_h}{\Delta_h} + 1 \right).$$

Вероятность столкновений с препятствием снижается при увеличении значений параметров FOV_h, FOV_v, N, FPS и уменьшении значений параметров Δ_h, Δ_v (для уменьшения уровня дискретности среды). Но справедливы и выражения:

$$FOV_h = \Delta_h n_{h1}; \; FOV_v = \Delta_v n_v ;$$

$$FPS = \frac{1}{\Delta_h n_{h1} \left(n_v + 1 \right) + \Delta_v n_v}.$$

Таким образом, возникает проблема поиска оптимальных значений шагов Δ_h, Δ_v и их количества n_{h1}, n_v. Решение этой проблемы является темой отдельного исследования и в рамках данной работы не рассматривалось. Использовались значения параметров, определённые опытным путём:

$\Delta_h = \Delta_v = 2°;$

$n_{h1} = 24; \; n_v = 4;$

$FOV_h = 48°; \; FOV_v = 8°;$

$FPS = 3\,\text{Гц}$.

Значение FPS меньше теоретического из-за ограничений, наложенных синхронизацией публикации данных разработанного плагина дальномера и симуляционного

времени «Gazebo». Представленный наборов параметров является одним из возможных, с которым достигается стабильный полёт с обходом препятствий.

Среда моделирования и разработанное программное обеспечение

Поставленная задача предполагает полную симуляцию полёта, поэтому необходим проект полётного программного обеспечения, который позволяет провести моделирование по принципу SITL (software-in-the-loop, симуляция работы без использования реального аппаратного обеспечения).

Существует несколько полётных стеков с открытым исходным кодом программного и аппаратного обеспечения. Среди них выделяется проект Pixhawk и его оригинальный полётный стек PX4 [5; 6]. В рамках данного исследования важно отметить наличие в данном стеке необходимых функций – автономной навигации с обходом препятствий – стек PX4/avoidance [7]. Кроме того, проект интегрируется с операционной системой ROS (Robot Operation System, операционная система для роботов) и симулятором Gazebo [8; 9]. Всё это делает его подходящей площадкой для проведения модельного эксперимента.

ROS основана на архитектуре графов, где обработка данных происходит в узлах, которые могут получать и передавать сообщения между собой, используя топики (темы), которые в дальнейшем именуются «Поток данных». В роли пользователей выступают узлы, которые могут публиковать сообщения и/или читать их, подписываясь на определенные темы.

Стек PX4/avoidance [10], предоставляющий алгоритмы полёта БПЛА с обходом препятствий, опирается на информацию об окружении в формате облака точек – наборе вершин в трёхмерной системе координат. Главному пакету данного стека «local_planner» необходимо, чтобы существовал поток данных, в который регулярно поступает такая информация. Для выполнения задачи, поставленной в работе, разработаны:

– модель устройства для симулятора Gazebo в формате SDF (Simple Description Format, простой формат описания) и операционной системы ROS в формате URDF (Unified Robot Description Format, унифицированный формат описания робота);
– модель БПЛА – квадрокоптера с этим устройством в форматах SDF и URDF;
– плагин устройства для Gazebo, имитирующий лазерный дальномер, который бы передавал информацию в поток данных ROS;
– узел, управляющий моторами подвеса и определяющий логику сканирования;
– узел, преобразующий данные с устройства в облако точек.

Так как ROS основана на архитектуре графов, то процесс симуляции лучше всего представить виде графа (рис. 2).

На рис. 2 фигуры овальной формы – узлы, прямоугольной – потоки данных, белым фоном выделены сторонние компоненты. Стрелки с увеличенной толщиной показывают связи в первой ветви графа, с обычной – второй, с штриховой линией – третьей. Корневым узлом можно считать симулятор Gazebo. Плагин лазерного дальномера, который подгружается в Gazebo, публикует данные в поток данных /range. Эти данные используются в двух ветвях графа.

Первая ветвь отвечает за создание развёртки. Здесь решается задача изменения направления лазерного дальномера в Gazebo. Контроллер устройства с каждым новым сообщением в потоке данных /range публикует сообщение со значением угла в один из потоков данных для управления моторами. В свою очередь, плагин модели в Gazebo воспринимает эти потоки данных и выставляет углы управляемых соединений в необ-

ходимую позицию. Таким образом, направление лазерного дальномера постоянно изменяется.

Вторая ветвь отвечает за создание облака точек. Основной узел, который здесь работает – это узел, объединяющий данные с устройства, «sensor_pointcloud». Узел получает данные от дальномера из потока данных «/range», данные о статусе сканирования из потока «/release_data» и данные о трансформациях «/tf» (под «трансформацией» (англ. «transform data») будем понимать набор данных, включающих координаты и углы поворота объекта относительно другого объекта).

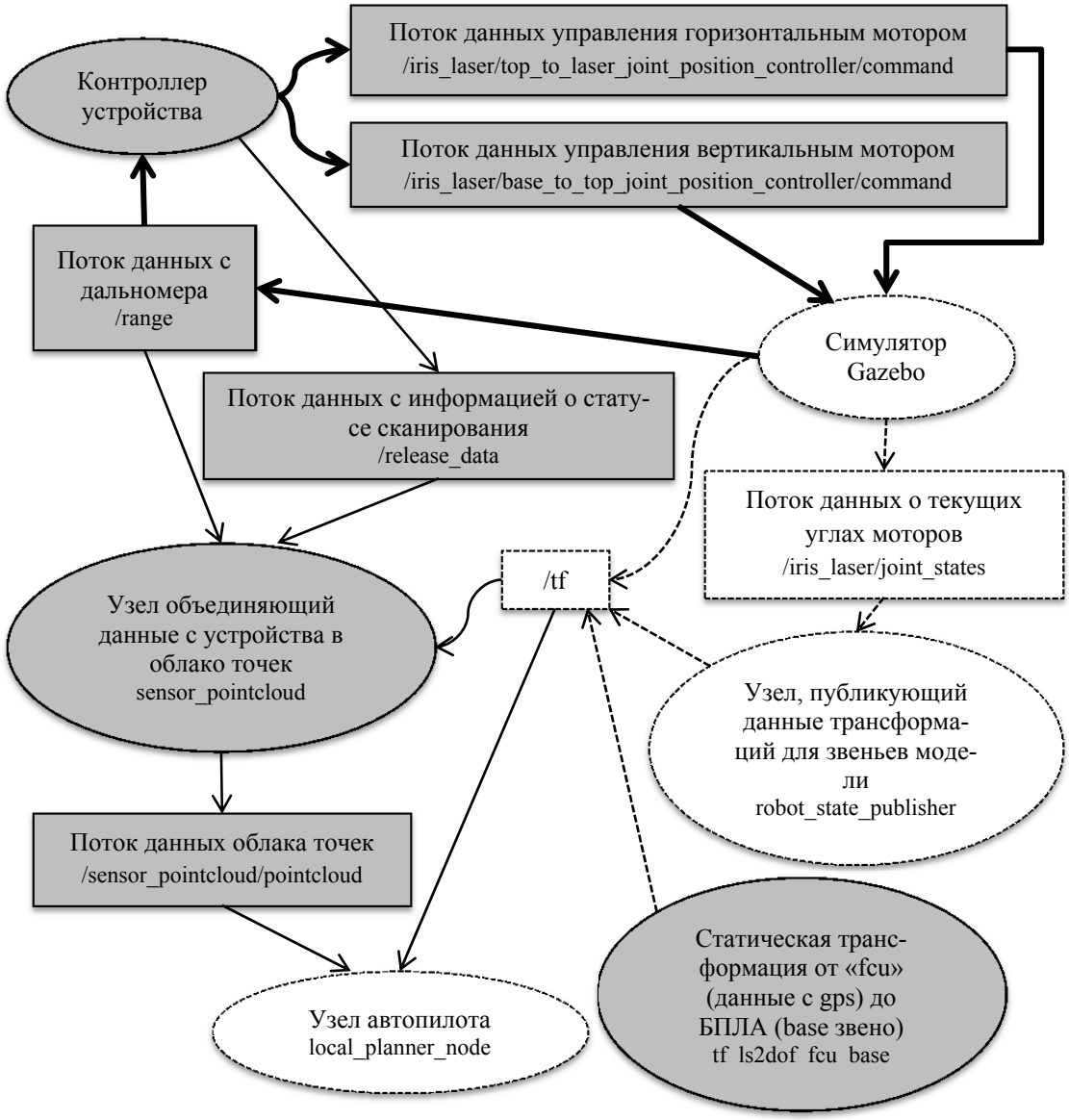

Рис. 2. Граф связей в процессе симуляции

К каждому сообщению «sensor_msgs: Range» из потока данных «/range», соответственно временному штампу, подбирается положение в пространстве лазерного дальномера. Имея эти данные, можно определить точку, на которую в данный момент направлен дальномер. Это может быть точка на препятствии, либо точка в свободном

пространстве, так как дальномер возвращает значение максимальной дистанции по умолчанию. В процессе накопления и объединения полученных данных создается буферное облако точек.

Так как наполнение облака точек происходит по ходу поворота моторов, облако точек заполняется постепенно из одной области к другой. Это может приводить к неправильной работе алгоритма автономной навигации (узел «local_planner_node») и принятию им неверных решений по управлению БПЛА. Вследствие этого принято решение публиковать только полные развёртки. Для этого используется поток данных «/release_data», в который контроллер устройства сообщает статус сканирования. Когда поступает сообщение, что выполнена полная развёртка, узел «sensor_pointcloud» чистит облако точек – защёлку, потом снова его заполняет, копируя данные из буферного облака точек. Затем буферное облако точек очищается, так как алгоритм локальный и глобальная карта не строится. Полученное облако точек – защёлка теперь будет публиковаться с частотой 20 раз в секунду в поток данных «/sensor_pointcloud/pointcloud» без изменений до момента, пока снова не будет выполнена полная развёртка и не придёт соответствующее сообщение от контроллера устройства в поток «/release_data». Поток данных «/sensor_pointcloud/pointcloud» является «входной» информацией для узла «local_planner_node», который является частью стека «PX4/Avoidance» и является самостоятельной реализацией алгоритма локальной навигации с обходом препятствий. Этот узел генерирует управляющие команды для квадрокоптера (главным образом, путевые точки).

Третья ветвь графа отвечает за трансформации. Данные о трансформациях всех звеньев модели находятся в одном потоке данных «/tf». Определение нужного отношения в каждом случае происходит по данным: трансформация от какого звена к какому звену и временному штампу. По этим данным всегда можно построить дерево трансформаций и определить координаты и ориентацию всех звеньев модели, для которых какой-либо узел публикует данные.

Результаты модельного эксперимента

Для визуализации результатов модельного эксперимента используется специальный инструмент визуализации ROS. Выбран вид сверху, т.к. на нём хорошо видна траектория полёта.

На первом снимке (рис. 3) представлена первая траектория полёта из начальной точки в целевую. В данном тестовом полете БПЛА успешно обошёл препятствия. Траектории следующих трёх тестовых полётов представлены на рис. 4.

На рисунке траектория представлена двумя кривыми, которые на большей части своей протяжённости совпадают. Одна линия отражает реальную траекторию полёта, другая опережает первую в процессе симуляции и отражает предполагаемую дальнейшую траекторию движения. В точках сближения с препятствием предполагаемая траектория доходит до самого препятствия, а реальная – нет, так как БПЛА останавливается на определённом расстоянии до препятствия и начинает искать другую траекторию, исключающую столкновение. Минимальная дистанция между БПЛА и препятствием в таких случаях составляет 1…1,5 м, а определение возможности столкновения происходит на расстоянии порядка 8 м до точки столкновения.

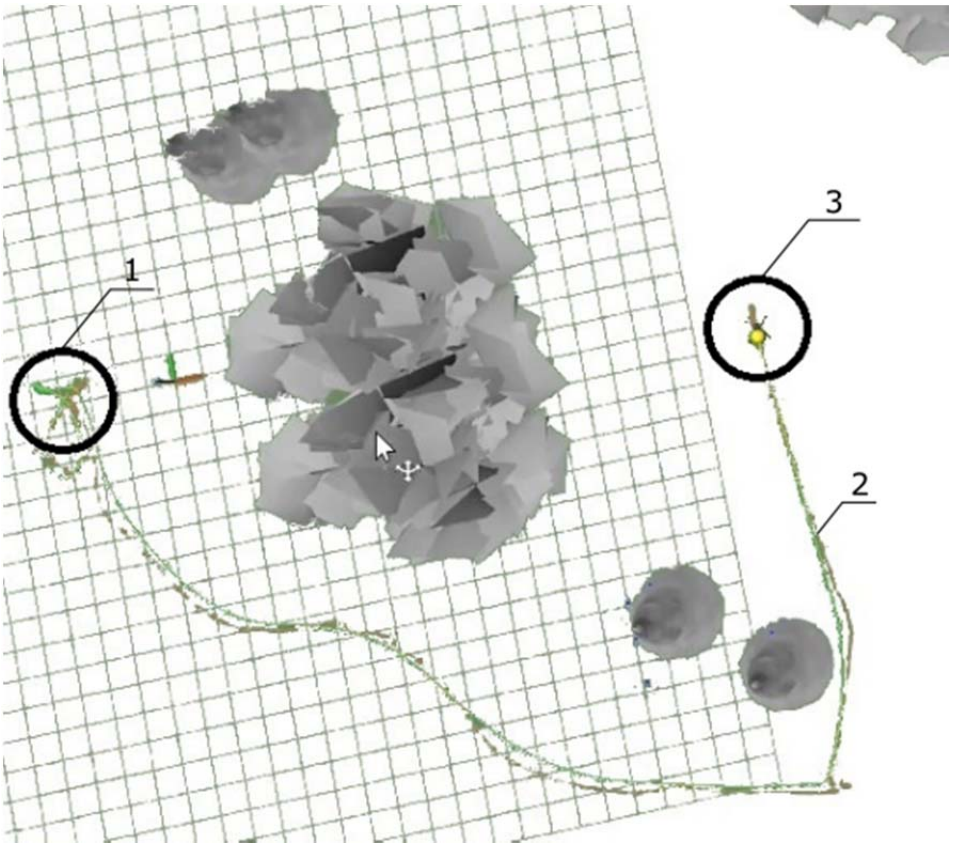

Рис. 3. Результаты модельного эксперимента – траектория полёта:
1 – начальная точка; 2 – траектория; 3 – целевая точка

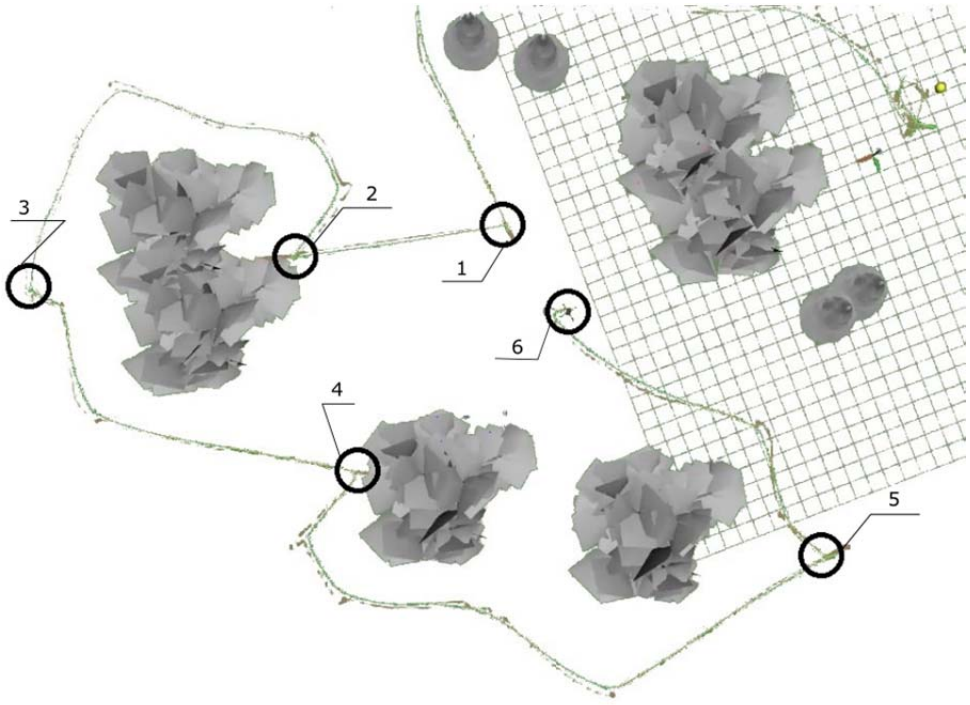

Рис. 4. Результаты модельного эксперимента – траектория полёта:
1 – начальная точка траектории; 3, 5 – промежуточные целевые точки;
2, 4 – точки опасного сближения с препятствием; 6 – конечная

Причиной этого являются небольшое количество полных циклов сканирований в секунду FPS, что было рассмотрено выше, а также особенности программного обеспечения «PX4/Avoidance». Данный стек разрабатывался для использования в первую очередь стереокамер, как датчиков окружения, и предполагает получение в качестве исходных данных значительно более плотно заполненное облако точек. Кроме этого, БПЛА при движении вперёд наклоняется. При этом меняется область обзора, что не учтено при проектировании логики контроллера устройства. Вследствие этого препятствие, лежащее на пути, обнаруживается позже. Влияние этих проблем можно минимизировать с помощью модернизации алгоритма – реализовать адаптивность размера области обзора в зависимости от текущей скорости БПЛА и добавить коррекцию значений вертикальных углов направления луча в зависимости от угла наклона БПЛА.

Перечень оборудования и ПО для использования предлагаемого устройства на реальном БПЛА

Для функционирования всех программных пакетов необходима производительность процессора не менее 36 гигафлопсов, объем ОЗУ не менее 2 Гбайт. В качестве «компьютера-компаньона» можно использовать такие устройства, как Intel NUC, Jetson TX2, Intel Atom x7-Z8750. Это полностью согласуется со списком моделей, на которых был протестирован «PX4/Avoidance». На данном компьютере должна быть установлена операционная система ROS и операционная система, совместимая с ROS. Оптимальным решением будет использование ОС «Ubuntu server 18.04» с дистрибутивом «ROS Melodic». В качестве полётного контроллера для БПЛА целесообразно использовать «Pixhawk». Данный контроллер позволяет подключать дальномеры напрямую, поэтому для реализации устройства будет достаточно дальномера [4], двух сервомоторов, подвеса и фиксирующей рамы.

Заключение

В работе представлен и протестирован с помощью модельного эксперимента сканирующий лазерный дальномер на подвесе с двумя степенями свободы. Данное устройство может быть использовано для реализации функций обхода препятствий и предотвращения столкновений БПЛА.

В процессе модельного эксперимента БПЛА последовательно проследовал к четырём целевым точкам, облетая препятствия. Выявлена проблема заметного сближения БПЛА с препятствием при наборе скорости. Установлено, что при работе с рассматриваемым устройством алгоритм автономной навигации «PX4/Avoidance» при движении с максимально допустимой скоростью 3 м/с определяет наличие препятствия на расстоянии порядка 8 м до точки возможного столкновения. При этом до остановки БПЛА пролетает ещё около 6,5…7,0 м. Таким образом, БПЛА останавливается только на расстоянии 1,0…1,5 м до препятствия. Для решения этой проблемы необходимо изменить алгоритм так, чтобы информация о точках, лежащих в области направления движения, обновлялась в реальном времени, а область обзора адаптировалась в зависимости от наклона и поворота БПЛА. Другой вариант решения проблемы – добавить дополнительный датчик в СТЗ, который будет обеспечивать информацией о препятствиях лишь в той области, в которую движется БПЛА, но с большей частотой обновления этой информации.

Несмотря на описанную выше проблему, сканирующий лазерный дальномер, закреплённый на подвесе с двумя степенями свободы, обеспечивает удовлетворительную навигацию БПЛА с возможностью обхода препятствий.

Библиографический список

1. Puck Lidar Sensor, High-Value Surround Lidar. Velodyne Lidar. https://velodynelidar.com/products/puck/

2. Leddar Vu8 Solid-State LiDAR https://leddartech.com/solutions/leddar-vu8-solid-state-lidar-sensor-module/

3. LeddarTech Vu 8 Channel Module, 99°/3° FOV, USB, CAN Bus, RS485 & UART – RobotShop. https://www.robotshop.com/en/leddartech-vu-8-channel-module-99-3-fov-usb-can-bus-rs485--uart.html

4. LIDAR-Lite 3 Laser Rangefinder High Performance (LLV3HP) – RobotShop. https://www.robotshop.com/en/lidar-lite-3-laser-rangefinder-high-performance-llv3hp.html

5. Pixhawk. The hardware standard for open-source autopilots. https://pixhawk.org/

6. Open Source Autopilot for Drones – PX4 Autopilot. https://px4.io/

7. PX4/avoidance – PX4 avoidance ROS node for obstacle detection and avoidance. https://github.com/PX4/avoidance

8. ROS. https://www.ros.org/

9. Gazebo. http://gazebosim.org/

10. Baumann T. Obstacle Avoidance for Drones Using a 3DVFH* Algorithm. Master Thesis. Zurich, 2018. 67 p.

LASER-SCANNING RANGEFINDER FIXED ON A GIMBAL WITH TWO DEGREES OF FREEDOM

V. A. Zelenskiy Doctor of Science (Engineering), Associate Professor, Professor of the Department of Design and Technology of Electronic Systems and Devices;
Samara National Research University, Samara, Russian Federation;
vaz-3@yandex.ru

M. V. Kapalin Postgraduate Student of the Department of Design and Technology of Electronic Systems and Devices;
Samara National Research University, Samara, Russian Federation;
no95typem@yandex.ru

A laser-scanning rangefinder mounted on a gimbal with two degrees of freedom is presented. The rangefinder can be used as part of a navigation system of an unmanned aerial vehicle to avoid obstacles or prevent collisions. Compared to stereo cameras, the device requires significantly less computing resources and is less dependent on lighting conditions. Compared to integrated lidars, the cost of the device is by an order lower. A model of the device was developed and an obstacle avoidance flight was simulated in the Gazebo simulator. The PX4/Avoidance software was used as an autopilot. As a result of a model experiment, we found that a scanning laser rangefinder can provide autonomous navigation with obstacle avoidance.

Laser-scanning rangefinder; gimbal with two degrees of freedom; unmanned aerial vehicle; obstacle avoidance; PX4/Avoidance autopilot; Gazebo simulator

Citation: Zelenskiy V.A., Kapalin M.V. Laser-scanning rangefinder fixed on a gimbal with two degrees of freedom. *Vestnik of Samara University. Aerospace and Mechanical Engineering*. 2021. V. 20, no. 3. P. 37-48.
DOI: 10.18287/2541-7533-2021-20-3-37-48

References

1. Puck Lidar Sensor, High-Value Surround Lidar. Velodyne Lidar. Available at: https://velodynelidar.com/products/puck/

2. Leddar Vu8 Solid-State LiDAR. Available at: https://leddartech.com/solutions/leddar-vu8-solid-state-lidar-sensor-module/

3. LeddarTech Vu 8 Channel Module, 99°/3° FOV, USB, CAN Bus, RS485 & UART – RobotShop. Available at: https://www.robotshop.com/en/leddartech-vu-8-channel-module-99-3-fov-usb-can-bus-rs485--uart.html

4. LIDAR-Lite 3 Laser Rangefinder High Performance (LLV3HP) – RobotShop. Available at: https://www.robotshop.com/en/lidar-lite-3-laser-rangefinder-high-performance-llv3hp.html

5. Pixhawk. The hardware standard for open-source autopilots. Available at: https://pixhawk.org/

6. Open Source Autopilot for Drones – PX4 Autopilot. Available at: https://px4.io/

7. PX4/avoidance - PX4 avoidance ROS node for obstacle detection and avoidance. Available at: https://github.com/PX4/avoidance

8. ROS. Available at: https://www.ros.org/

9. Gazebo. Available at: http://gazebosim.org/

10. Baumann T. Obstacle Avoidance for Drones Using a 3DVFH* Algorithm. Master Thesis. Zurich, 2018. 67 p.