МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Факультет информационных технологий

Кафедра компьютерных технологий

Направление подготовки: 09.03.01 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

Образовательная программа: 09.03.01 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

РАЗРАБОТКА УПРАВЛЕНИЯ ДВИЖЕНИЕМ КВАДРОКОПТЕРА ПО ЗАДАННОЙ ТРАЕКТОРИИ

утверждена распоряжением проректора по учебной работе №134 от «04» мая 2017 г.

Грачев Виталий Игоревич, группа 14202 _____

(Фамилия, Имя, Отчество студента, группа) (подпись студента)

«К защите допущена»

Заведующий кафедрой,

д.т.н, доцент

Зюбин В. Е./_____

(ФИО) / (подпись)

«……»……2018 г.

Руководитель ВКР

д.т.н., доцент

зав. кафедрой КТ ФИТ НГУ

Зюбин В. Е./_____

(ФИО) / (подпись)

«……»……2018 г.

Дата защиты: «……»……2018 г.

Новосибирск, 2018 г.

МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Факультет информационных технологий

Кафедра компьютерных технологий

НАПРАВЛЕНИЕ ПОДГОТОВКИ: 09.03.01 Информатика и вычислительная техника

УТВЕРЖДАЮ

Зав. кафедрой Зюбин В. Е.
(Фамилия, И., О.)
…….
(подпись)
«…»……20…г.

ЗАДАНИЕ

НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА

Студенту Грачеву Виталию Игоревичу, группы 14202
(фамилия, имя, отчество, номер группы)
Тема: Разработка управления движением квадрокоптера по заданной траектории

утверждена распоряжением проректора по учебной работе от «04» мая 2017 г. № 134

Срок сдачи студентом готовой работы: 31 мая 2018 г.

Исходные данные (или цель работы): Реализация алгоритма формирования управляющих воздействий в задаче движения квадрокоптера по заданной траектории.

Структурные части работы: Введение, анализ предметной области и требования, проектирование программного решения, реализация программного решения, практическая апробация.

Руководитель ВКР
заведующий кафедрой КТ ФИТ НГУ,
д.т.н., доцент
Зюбин В. Е./_____
(ФИО) / (подпись)
«…»……20…г.

Задание принял к исполнению
Грачев В. И./_____
(ФИО студента) / (подпись)
«…»……20…г.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ 4
ВВЕДЕНИЕ 5
Глава 1 Анализ и требования 7
1.1 Обзор реализуемого подхода 7
1.1.1 Объект управления 7
1.1.2 Задание траектории движения 8
1.1.3 Обеспечение движения по траектории 9
1.2 Требования 10
1.3 Выбор бортового ПО 11
Глава 2 Проектирование 13
2.1 Архитектура программного решения 13
2.2 Проектирование контроллера траекторного управления 13
2.3 Проектирование адаптера 14
Глава 3 Реализация 16
3.1 Выбор языка программирования 16
3.2 Контроллер траекторного управления 16
3.2.1 Реализация контроллера позиции 16
3.2.2 Реализация контроллера ориентации 18
3.2.3 Реализация контроллера траекторного управления 18
3.3 Адаптер 18
3.3.1 Реализация модуля траекторного управления 18
3.3.2 Модификация модуля commander 20
Глава 4 Практическая апробация 21
4.1 Численное моделирование 21
4.2 Практические испытания 22
4.2.1 Испытательный стенд 22
4.2.2 Апробация контроллера ориентации 23
4.2.3 Апробация контроллера траекторного управления 25
ЗАКЛЮЧЕНИЕ 27
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ 28

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

ПИД – пропорционально-интегрально-дифференцирующий.

ИАиЭ – институт автоматики и электрометрии.

СО РАН – Сибирское отделение Российской академии наук.

ПО – программное обеспечение.

Бортовое ПО – программное обеспечение бортовой ЭВМ.

GPS – спутниковая система навигации, определяющая местоположение во всемирной системе координат.

ВВЕДЕНИЕ

К преимуществам квадрокоптера можно отнести: простоту конструкции, способность зависать в воздухе на одном месте, вертикальные взлет и посадку, высокую маневренность. Эти преимущества делают его применение привлекательным в задачах транспортировки грузов, фото- и видеосъемки, наблюдения за состоянием окружающей среды в неблагоприятных для человека условиях. Эти задачи требуют повышенной точности маневров, в особенности внутри помещений, где свободное пространство ограничено.

В публикациях по вопросам траекторного управления обсуждаются различные подходы, в частности, использование линейных ПИД-регуляторов [1], скользящего режима [2], контроллера на основе нейронной сети [3].

Сотрудниками лаборатории №9 ИАиЭ СО РАН был предложен подход к организации движения квадрокоптера в пространстве состояний на основе метода структурного синтеза [4]. Особенностью данного подхода в сравнении с вышеперечисленными является известное гарантированное время переходного процесса [5]. Однако данный подход еще не имеет полноценной практической реализации.

Цель работы – реализация алгоритма формирования управляющих воздействий в задаче движения квадрокоптера по заданной траектории.

Для достижения поставленной цели был выделен следующий набор задач:

1) проанализировать объект управления и специфику реализуемого подхода;
2) сформулировать требования к разрабатываемому ПО;
3) спроектировать программное решение;
4) реализовать программное решение;
5) произвести практическую апробацию программного решения.

Работа состоит из четырех глав. В первой главе приведен анализ объекта управления и специфики реализуемого подхода, а также сформулированы требования. Вторая глава посвящена вопросам проектирования программного

решения. В третьей главе описывается реализация программного решения. В четвертой главе изложены результаты практической апробации.

Глава 1 Анализ и требования

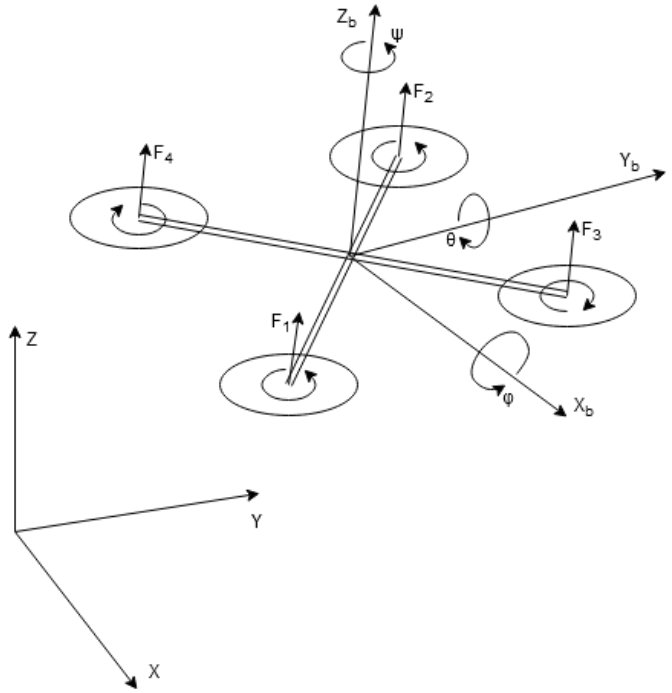
1.1 Обзор реализуемого подхода

1.1.1 Объект управления

Движение квадрокоптера описывается системой из шести нелинейных дифференциальных уравнений [6]:

$$\begin{cases} m \cdot \ddot{x} = (sin\,\psi \cdot sin\,\varphi + cos\,\psi \cdot cos\,\varphi \cdot sin\,\theta) \cdot u_{thrust} \\ m \cdot \ddot{y} = (-cos\,\psi \cdot sin\,\varphi + sin\,\psi \cdot cos\,\varphi \cdot sin\,\theta) \cdot u_{thrust} \\ m \cdot \ddot{z} = cos\,\varphi \cdot cos\,\theta \cdot u_{thrust} - m \cdot g \\ I_{xx} \cdot \ddot{\varphi} = u_{x_b} - (I_{zz} - I_{yy}) \cdot \dot{\theta} \cdot \dot{\psi} \\ I_{yy} \cdot \ddot{\theta} = u_{y_b} - (I_{xx} - I_{zz}) \cdot \dot{\varphi} \cdot \dot{\psi} \\ I_{zz} \cdot \ddot{\psi} = u_{z_b} \end{cases}$$

Здесь x, y, z – координаты центра масс квадрокоптера в неподвижной декартовой системе координат, φ, θ, ψ – углы Эйлера, описывающие ориентацию квадрокоптера, m – масса квадрокоптера, g – ускорение, создаваемое силой тяжести, I_{xx}, I_{yy}, I_{zz} – моменты инерции относительно соответствующих осей x_b, y_b, z_b связанной с квадрокоптером системы координат, $u_{x_b}, u_{y_b}, u_{z_b}$ – вращающие моменты относительно соответствующих осей квадрокоптера, а u_{thrust} – сила, направленная вдоль оси z_b. На рисунке 1 представлена схема модели квадрокоптера и упомянутые системы координат.



xyz – неподвижная система координат, $x_b y_b z_b$ – связанная с квадрокоптером система координат

Рисунок 1 – Схема модели квадрокоптера

Управляющие воздействия $u_{x_b}, u_{y_b}, u_{z_b}, u_{thrust}$ связаны с тягами двигателей следующим соотношением [5]:

$$\begin{bmatrix} u_{x_b} \\ u_{y_b} \\ u_{z_b} \\ u_{thrust} \end{bmatrix} = \begin{bmatrix} \frac{l}{\sqrt{2}} & -\frac{l}{\sqrt{2}} & -\frac{l}{\sqrt{2}} & \frac{l}{\sqrt{2}} \\ -\frac{l}{\sqrt{2}} & \frac{l}{\sqrt{2}} & -\frac{l}{\sqrt{2}} & \frac{l}{\sqrt{2}} \\ \lambda & \lambda & -\lambda & -\lambda \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix}$$

где l – расстояние от осей винтов до центра масс квадрокоптера, λ – коэффициент пропорциональности между тягами винтов и создаваемыми ими реактивными моментами вращения относительно оси z_b.

1.1.2 Задание траектории движения

Для описания желаемого характера движения используются следующие соотношения:

$f(x, y) = 0$ – задает желаемую траекторию в плоскости XY;

$z = z_{ref}(x, y)$ – задает желаемую высоту движения;

$v = v_{ref}\,(x, y)$ – задает желаемую скорость движения в плоскости XY.

При этом на данные соотношения накладываются следующие ограничения:

1) производные функции $f(x, y)$ существуют до второго порядка включительно;
2) первые производные не равны нулю одновременно;
3) кривая $f(x, y) = 0$ не имеет точек самопересечения, кроме, может быть, начала и конца;
4) функции $z_{ref}\,(x, y)$ и $v_{ref}\,(x, y)$ являются кусочно-постоянными.

1.1.3 Обеспечение движения по траектории

Пусть t_0 и t_1 моменты начала и конца движения квадрокоптера. Момент времени t такой, что $t \in [t_0, t_1]$. Функции $x(t), y(t), z(t)$ задают положение центра масс квадрокоптера в момент времени t.

Вводятся следующие функции, выражающие отклонение квадрокоптера от заданного характера движения в момент времени t:

$$F(t) = f(x(t), y(t));$$

$$Z(t) = z(t) - z_{ref}\,(x(t), y(t));$$

$$V(t) = v(t) - v_{ref}\,(x(t), y(t)).$$

Вводятся вспомогательные функции следующего вида:

$$S_f(t) = \frac{dF}{dt}(t) + k_f F(t);$$

$$S_z(t) = \frac{dZ}{dt}(t) + k_z Z(t);$$

$$S_v(t) = V(t).$$

Соотношения $\frac{dS_f}{dt}(t) = -\alpha_f S_f(t)$, $\frac{dS_z}{dt}(t) = -\alpha_z S_z(t)$, $\frac{dS_v}{dt}(t) = -\alpha_v S_v(t)$, где $\alpha_f, \alpha_z, \alpha_v$ – положительные постоянные, обеспечивают сходимость функций $S_f(t), S_z(t), S_v(t)$ к нулю.

Таким образом, гарантируется, что отклонение от заданного характера движения будет уменьшаться по следующему закону, зависящему от положительных постоянных $k_f, k_z, \alpha_f, \alpha_z, \alpha_v$:

$$F(t) = \frac{F(t_0)}{\alpha_f - k_f}\left(\alpha_f exp\left(-k_f(t-t_0)\right) - k_f exp\left(-\alpha_f(t-t_0)\right)\right);$$

$$Z(t) = \frac{Z(t_0)}{\alpha_z - k_z}\left(\alpha_z exp(-k_z(t-t_0)) - k_z exp(-\alpha_z(t-t_0))\right);$$

$$V(t) = V(t_0) exp(-\alpha_v(t-t_0)).$$

В [5] выводятся формулы вычисления ориентации $\varphi_{ref}(t), \theta_{ref}(t), \psi_{ref}(t)$ и суммарной тяги винтов u_{thrust}, поддержание которых обеспечивает движение квадрокоптера в пространстве, удовлетворяющее соотношениям $\frac{dS_f}{dt}(t) = -\alpha_f S_f(t)$, $\frac{dS_z}{dt}(t) = -\alpha_z S_z(t)$, $\frac{dS_v}{dt}(t) = -\alpha_v S_v(t)$.

Также, в [5] приводятся формулы вычисления управляющих воздействий $u_{x_b}(t), u_{y_b}(t), u_{z_b}(t)$, обеспечивающих принятие и поддержание квадрокоптером ориентации $\varphi_{ref}(t), \theta_{ref}(t), \psi_{ref}(t)$.

1.2 Требования

Для проведения практической апробации целесообразно построить программное решение на основе существующего бортового ПО с открытыми исходными кодами. В то же время, для упрощения повторного использования реализованного алгоритма на других программных и аппаратных платформах, следует избегать зависимости кода от выбранного бортового ПО.

В настоящее время в лаборатории №9 для большинства задач, связанных с управлением беспилотными летательными аппаратами, используется полетный контроллер Pixhawk [7]. По этой причине выбранное бортовое ПО должно быть совместимо с аппаратной платформой Pixhawk.

Чтобы проанализировать результаты практической апробации, нужно сравнить заданную траекторию с траекторией, по которой квадрокоптер

фактически двигался. Для этого необходимо сохранять информацию о положении и ориентации квадрокоптера во время полета.

Нельзя отрицать возможность совершения ошибок при реализации алгоритма. Некоторые из них могут вызвать непредсказуемое поведение квадрокоптера и привести к повреждению его или окружающего оборудования во время отладки. Во избежание такой ситуации необходима возможность оперативно переключить квадрокоптер в режим ручного управления с пульта дистанционного управления, чтобы безопасно посадить его.

Таким образом, к реализуемому решению предъявляются следующие требования:

1) программное обеспечение должно строиться на основе существующего бортового ПО;
2) используемое бортовое ПО должно иметь открытые исходные коды;
3) используемое бортовое ПО должно быть совместимо с Pixhawk;
4) реализация алгоритма не должна зависеть от используемого бортового ПО;
5) должно быть обеспечено логирование информации о положении и ориентации квадрокоптера во время полета;
6) должна быть обеспечена возможность оперативно переключить квадрокоптер в режим ручного управления с пульта дистанционного управления.

1.3 Выбор бортового ПО

В качестве бортового ПО для аппаратной платформы Pixhawk могут выступать:

1) PX4 [8];
2) ArduPilot [9].

PX4 и Pixhawk являются, соответственно, программной и аппаратной частью одного проекта – Dronecode [10]. По этой причине их совместимость

является естественной. В то же время, ArduPilot для обеспечения работы с Pixhawk использует PX4 как промежуточное ПО.

Разрабатываемый в данной работе алгоритм предполагает работу на низком уровне, потому, с целью исключения лишних уровней взаимодействия, было принято решение выбрать в качестве бортового ПО PX4.

Полезным свойством PX4 является встроенная система логирования на карту памяти информации об основных параметрах квадрокоптера во время полета, в том числе о положении и ориентации. Поэтому выбор PX4 в качестве бортового ПО автоматически обеспечивает удовлетворение требования 5.

Глава 2 Проектирование

2.1 Архитектура программного решения

С целью одновременного выполнения требований 1 и 4 было применено разделение программного решения на две части:

1) контроллер траекторного управления, не зависящий от специфики полетного контроллера, реализующий алгоритм управления квадрокоптером;
2) адаптер, обеспечивающий взаимодействие контроллера траекторного управления с PX4.

2.2 Проектирование контроллера траекторного управления

Контроллер траекторного управления составлен из двух частей:

1) контроллера позиции;
2) контроллера ориентации.

Контроллер позиции ответственен за расчет суммарной тяги вдоль оси z_b и ориентации, которую квадрокоптер должен принять, чтобы придерживаться заданного характера движения. Эта требуемая ориентация $(\varphi_{ref}, \theta_{ref}, \psi_{ref})$ передается контроллеру ориентации.

Контроллер ориентации ответственен за расчет вращающих моментов $(u_{x_b}, u_{y_b}, u_{z_b})$, которые необходимы для принятия требуемой ориентации, полученной от контроллера позиции.

Входные данные контроллера траекторного управления:

1) функция $f(x, y)$;
2) функция $z_{ref}(x, y)$;
3) функция $v_{ref}(x, y)$;
4) параметры алгоритма $k_f, k_z, \alpha_f, \alpha_z, \alpha_v, k_\varphi, k_\theta, k_\psi, \alpha_\varphi, \alpha_\theta, \alpha_\psi$;
5) текущее положение квадрокоптера (x, y, z);
6) текущие линейные скорости квадрокоптера $(\dot{x}, \dot{y}, \dot{z})$;
7) текущая ориентация квадрокоптера (φ, θ, ψ);

8) текущие угловые скорости квадрокоптера ($\dot{\varphi}, \dot{\theta}, \dot{\psi}$).

При этом значения пунктов 5-8 данного списка должны постоянно обновляться, чтобы поддерживать актуальное значение.

Выходные данные контроллера траекторного управления:

1) вектор управляющих воздействий ($u_{x_b}, u_{y_b}, u_{z_b}, u_{thrust}$).

Здесь $u_{x_b}, u_{y_b}, u_{z_b}$ – вращающие моменты вокруг соответствующих осей, а u_{thrust} – суммарная тяга винтов квадрокоптера.

2.3 Проектирование адаптера

При проектировании адаптера были изучены и приняты во внимание особенности PX4.

Бортовое программное обеспечение PX4 состоит из операционной системы реального времени, набора драйверов устройств и модулей, реализующих ту или иную функциональность. Каждый модуль решает одну задачу, исполняется в отдельном процессе и взаимодействует с другими модулями с помощью системы асинхронного обмена сообщениями uORB.

В соответствии с описанной выше концепцией был разработан модуль траекторного управления, в задачи которого входит обмен uORB-сообщениями с другими модулями с целью получения информации о текущем положении, ориентации, линейных и угловых скоростях, а также передачи вектора управляющих воздействий. Полученные данные подлежат передаче контроллеру траекторного управления, который, используя их, формирует вектор управляющих воздействий.

В PX4 квадрокоптер может находиться в разных режимах (MANUAL – ручного управления; AUTO_LOITER – зависания; POSCTL – управления позицией; OFFBOARD – удаленного управления; AUTO_MISSION – движения по маршруту, заданному ключевыми точками и других) [11], и в зависимости от текущего режима менять свое поведение.

Модуль commander в PX4 отвечает за манипуляции с текущим режимом: проверку допустимости перехода в другой режим, осуществление этого перехода и периодическую публикацию uORB-сообщения о текущем режиме и соответствующих режиму ограничениях на работу отдельных модулей.

Среди существующих режимов AUTO_MISSION является наиболее близким по смыслу к решаемой задаче движения по заданной траектории. Однако, в правилах перехода в этот режим обязательным условием является устойчивый сигнал GPS. Это значительно ограничивает возможность работы в данном режиме внутри помещений.

По этой причине было принято решение ввести дополнительный режим AUTO_TRAJECTORY для движения по заданной траектории, независимый от наличия сигнала GPS, что потребовало внесения соответствующих изменений в модуль commander.

Глава 3 Реализация

3.1 Выбор языка программирования

Бортовое программное обеспечение PX4 реализовано на языках C и C++, поэтому естественным решением было произвести реализацию на одном из этих языков.

Реализация производилась на языке C++, так как заложенный в нем объектно-ориентированный подход позволяет естественно выразить выделенные на этапе проектирования сущности.

3.2 Контроллер траекторного управления

3.2.1 Реализация контроллера позиции

Для реализации функциональности контроллера позиции был выделен класс PositionController. Его конструктор принимает объекты, описывающие требуемый характер движения, параметры алгоритма и массу квадрокоптера. PositionController предоставляет следующие методы:

- void update(const Vec3f & position, const Vec3f & velocity) – используется для внесения в контроллер информации об актуальном положении и линейных скоростях квадрокоптера;
- void set_movement_params(const MovementParams & params) – используется для изменения характера движения;
- void set_algorithm_params(const AlgorithmParams & params) – используется для изменения параметров алгоритма;
- const Vec3f & get_ref_angles() – используется для получения рассчитанной контроллером требуемой ориентации;
- float get_thrust() – используется для получения рассчитанной контроллером требуемой суммарной тяги винтов квадрокоптера.

Расчеты, выполняемые контроллером позиции, имеют достаточно высокую вычислительную сложность, поэтому для сохранения быстродействия

важно избежать перерасчетов, не являющихся необходимыми. Для достижения этого были применены следующие техники:

- **ленивые вычисления** – перерасчет выполняется только тогда, когда требуется получить выходное значение, в противоположность подходу, когда перерасчет выполняется при изменении входных данных;
- **меморизация** – результаты расчетов сохраняются и сопровождаются флагом, сигнализирующим, действительны ли они. После расчета, результаты помечаются как действительные. При последующих обращениях к выходному значению, если сохраненное значение помечено как действительное, возвращается сохраненное значение, а перерасчет не производится. При изменении входных данных сохраненные значения помечаются как недействительные, что вызовет перерасчет при обращении к выходному значению.

Для удобства использования функции $f(x, y)$, описывающей траекторию движения, был выделен класс Trajectory. Для получения значений функций $f, \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial^2 f}{\partial x^2}, \frac{\partial^2 f}{\partial x \partial y}, \frac{\partial^2 f}{\partial y^2}$ в точке pos, заданной координатами x, y, класс Trajectory предоставляет соответствующие методы:

- float operator()(const Vec2f & pos);
- float d_dx(const Vec2f & pos);
- float d_dy(const Vec2f & pos);
- float d2_dx2(const Vec2f & pos);
- float d2_dxdy(const Vec2f & pos);
- float d2_dy2(const Vec2f & pos).

Конструктор класса Trajectory принимает от пользователя набор соответствующих функций, лямбда-выражений, выражений связывания или других функциональных объектов, для сохранения которых применяется универсальная обертка std::function. Для расчета каждого значения вызывается метод operator() соответствующего функционального объекта с заданным аргументом pos.

3.2.2 Реализация контроллера ориентации

Для реализации функциональности контроллера позиции был выделен класс AttitudeController.

AttitudeController предоставляет следующие методы:

- void update(const Vec3f & angles, const Vec3f & angular_velocity) – используется для внесения в контроллер информации об актуальной ориентации и угловых скоростях квадрокоптера;
- void set_algorithm_params(const AlgorithmParams & params) – используется для изменения параметров алгоритма;
- const Vec3f & get_u_values() – используется для получения рассчитанных контроллером требуемых вращающих моментов $u_{x_b}, u_{y_b}, u_{z_b}$.

3.2.3 Реализация контроллера траекторного управления

Контроллер траекторного управления реализован в виде класса TrajectoryController, агрегирующего в себе PositionController и AttitudeController.

TrajectoryController предоставляет следующие методы:

- const Vec4f & get_controls() – используется для получения вектора управляющих воздействий $(u_{x_b}, u_{y_b}, u_{z_b}, u_{thrust})$, который формируется из значений, предоставленных методом get_u_values() контроллера ориентации и методом get_thrust() контроллера позиции.

3.3 Адаптер

3.3.1 Реализация модуля траекторного управления

Работу модуля траекторного управления можно разделить на следующие этапы:

1) получение актуальной информации о текущих позиции, ориентации и ограничениях текущего режима;
2) проверка ограничений текущего режима;

3) передача информации о текущих позиции, ориентации, угловых и линейных скоростях контроллеру траекторного управления;
4) получение вектора управляющих воздействий от контроллера траекторного управления;
5) отправка рассчитанных управляющих воздействий модулю FMU, реализующему конвертацию в значения тяг винтов.

Первый этап представляет из себя чтение свежих uORB-сообщений vehicle_local_position, vehicle_attitude, vehicle_control_mode, содержащих информацию о текущих позиции, ориентации и ограничениях текущего режима соответственно.

Получение свежего сообщения состоит из:

1) проверки, что имеется новое, непрочитанное uORB-сообщение с помощью функции orb_check, предоставленной библиотекой системы асинхронного обмена сообщениями uORB;
2) копирования свежего сообщения в локальный буфер с помощью функции orb_copy, также предоставленной библиотекой системы асинхронного обмена сообщениями uORB.

На втором этапе производится проверка, разрешено ли управление роторами квадрокоптера данному модулю. Если нет, последующие этапы не выполняются.

На третьем этапе информация из поступивших свежих uORB-сообщений vehicle_local_position и/или vehicle_attitude передается контроллеру траекторного управления с помощью методов linear_update и angular_update соответственно.

На четвертом этапе осуществляется получение от контроллера траекторного управления рассчитанных управляющих воздействий с помощью метода get_controls.

На пятом этапе осуществляется отправка uORB-сообщения actuator_controls_0, содержащего управляющие воздействия.

3.3.2 Модификация модуля commander

В модуль commander были внесены изменения:

1) добавлен режим AUTO_TRAJECTORY;
2) добавлено правило перехода в режим AUTO_TRAJECTORY по переключению тумблера на пульте дистанционного управления, что обеспечивает удовлетворение требования 6;
3) добавлены соответствующие режиму AUTO_TRAJECTORY ограничения, запрещающие стандартным модулям, отвечающим за управление движением квадрокоптера, управление роторами квадрокоптера, и разрешающие осуществлять это управление модулю траекторного управления.

Глава 4 Практическая апробация

Для упрощения и ускорения процесса поиска ошибок было решено разделить апробацию на три этапа:

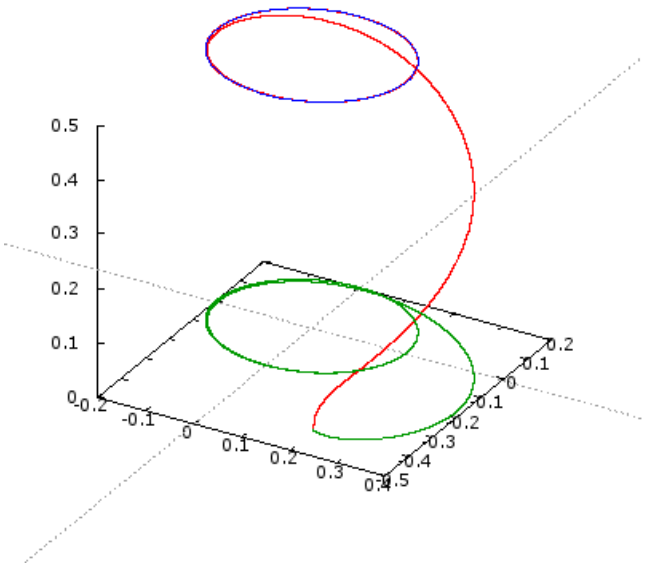
1) численное моделирование;
2) апробация контроллера ориентации;
3) апробация контроллера траекторного управления.

4.1 Численное моделирование

Для проведения численного моделирования была создана программа-симулятор на языке C++, реализующая модель квадрокоптера, описанную в подразделе 1.1.1 данной работы. Для осуществления управления моделью в программу-симулятор был встроен контроллер траекторного управления.

В процессе испытаний моделировались запуски квадрокоптера из различных точек, с различными заданными траекториями и наборами параметров алгоритма $k_f, k_z, \alpha_f, \alpha_z, \alpha_v, k_\varphi, k_\theta, k_\psi, \alpha_\varphi, \alpha_\theta, \alpha_\psi$.

На рисунке 2 представлен результат численного моделирования.



Синим цветом обозначена заданная траектория (окружность радиуса 20 см на высоте 50 см); красным цветом обозначена траектория движения модели квадрокоптера; зеленым цветом обозначена проекция траектории движения модели на плоскость XY.

Рисунок 2 – Результат моделирования

В ходе численного моделирования было выявлено, что не все комбинации входных данных способны привести к ожидаемому результату: если траектория подразумевает маневры с достаточно малым радиусом кривизны, то при значениях $v_{ref}(x, y)$ бóльших некото рого значения, модель становится неспособна выйти на заданную траекторию.

Для исправления этой ситуации следует дополнить подход алгоритмом адаптивного изменения $v_{ref}(x, y)$ в зависимости от радиуса кривизны траектории.

4.2 Практические испытания

4.2.1 Испытательный стенд

Испытания проводились на реальном квадрокоптере в помещении. Для получения координат и линейных скоростей квадрокоптера использовалась система, состоящая из внешней камеры на штативе, компьютера, оснащенного набором программ из пакета Robot Operating System (ROS) для определения пространственных координат маркера и закрепленного на квадрокоптере маркера. Рисунок 3 иллюстрирует закрепленный на квадрокоптере маркер.

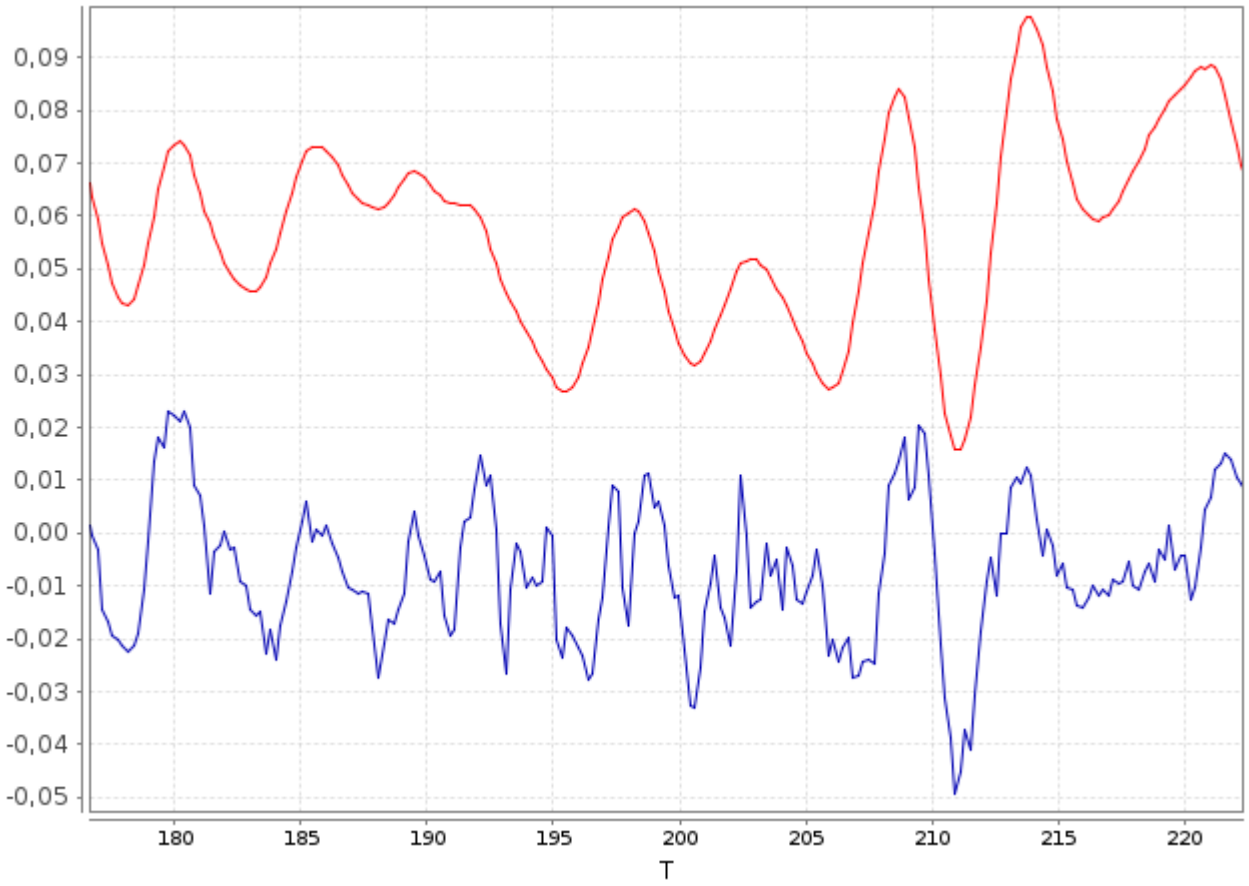
Рисунок 3 – Маркер для системы визуального позиционирования, закрепленный на квадрокоптере

Рассчитанные координаты передаются на квадрокоптер через WI-FI по протоколу Mavlink.

4.2.2 Апробация контроллера ориентации

Целью этого этапа было удостовериться, что контроллер ориентации успешно выполняет свою функцию установления заданных углов. Для упрощения анализа поведения квадрокоптера, контроллер позиции был временно заменен на более предсказуемый контроллер, реализующий пропорциональный регулятор, реализованный в виде класса, предоставляющего набор методов, аналогичный набору, предоставляемому классом PositionController.

В ходе апробации контроллера ориентации было обнаружено, что на реальном квадрокоптере, обладающем неидеальной центровкой, контроллер ориентации обеспечивает ненулевую статическую ошибку регулирования. Рисунок 4 иллюстрирует наличие статической ошибки регулирования.



Зависимость от времени требуемого угла тангажа (красным цветом) и фактического угла тангажа (синим цветом)

Рисунок 4 – Результат испытания контроллера ориентации, иллюстрирующий наличие статической ошибки регулирования

Это связано с тем, что контроллер ориентации имеет структуру, схожую с пропорционально-дифференцирующим регулятором, который не способен компенсировать действие не учтенной в модели внешней силы.

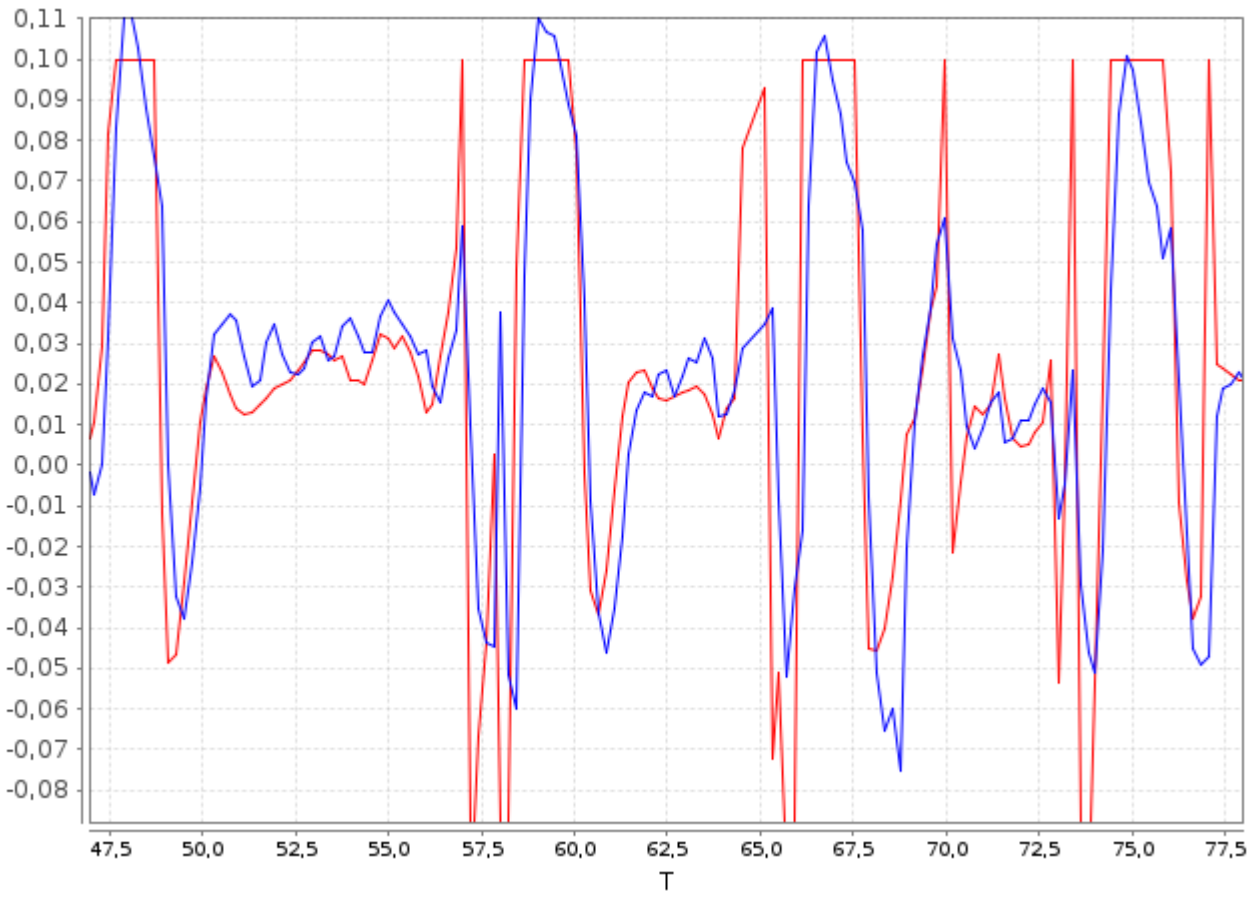
Были предложены следующие методы борьбы с ненулевой статической ошибкой:

1) добавление интегрального звена в контроллер ориентации;
2) предполетная регулировка – смещение значений выдаваемых управляющих воздействий $u_{x_b}, u_{y_b}, u_{z_b}$ для компенсации внешней силы.

Хотя метод 1 является более универсальным, добавление интегрального звена повышает порядок системы и может привести к неустойчивости.

Метод 2 является менее универсальным, так как не может подстраиваться под меняющиеся внешние силы. В условиях помещения отсутствует ветер, поэтому нет необходимости компенсировать переменчивую внешнюю силу, действующую на квадрокоптер. В таких условиях метод 2 является более подходящим вариантом, поэтому был применен в данной работе.

На рисунке 5 изображен результат испытания после предполетной регулировки.



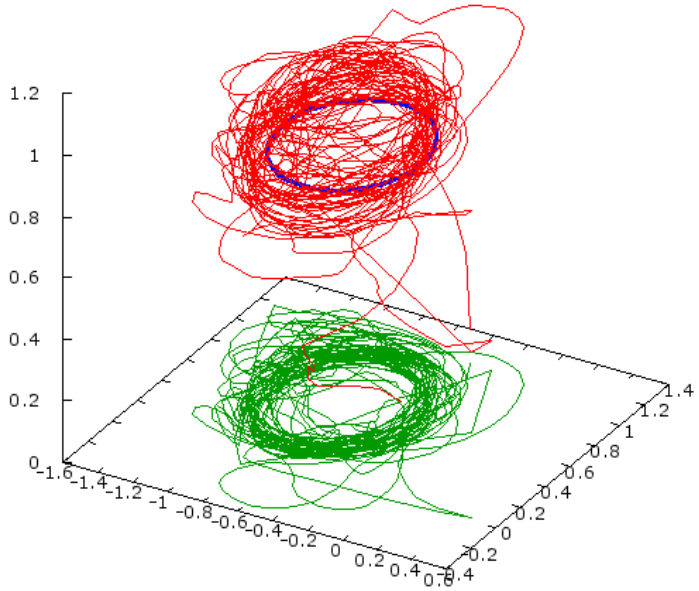
Зависимость от времени требуемого угла тангажа (красным цветом) и фактического угла тангажа (синим цветом)

Рисунок 5 – Результат испытания контроллера ориентации после предполетной регулировки

4.2.3 Апробация контроллера траекторного управления

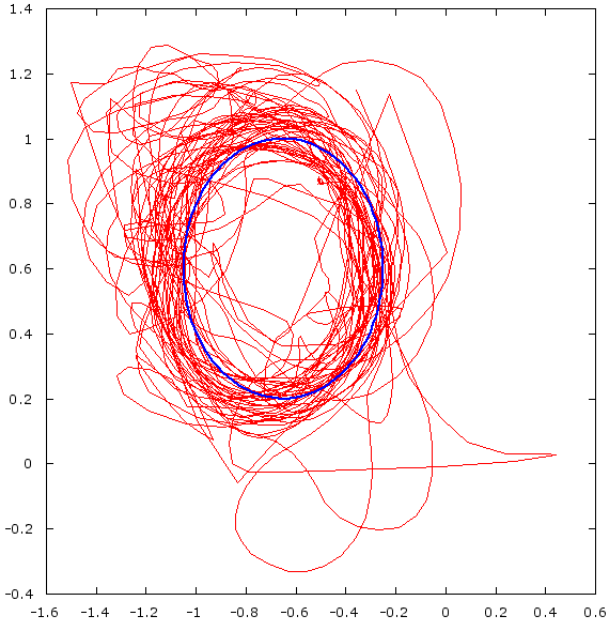
Испытания контроллера траекторного управления производились внутри помещения. Перед запусками производилась предполетная регулировка, предложенная по результатам испытаний контроллера позиции.

На рисунках 6 и 7 приведены результаты эксперимента при следующих параметрах: траектория – окружность радиусом 0,4 м, описанная вокруг точки с координатами $x = -0.65$м $y = 0.6$м; $z_{ref}(x, y) = 0.85$м; $v_{ref}(x, y) = 0{,}3$м; $k_f = 2.06; k_z = 1.3; \alpha_f = 2.07; \alpha_z = 1.31; \alpha_v = 3; k_\varphi = k_\theta = k_\psi = 5; \alpha_\varphi = \alpha_\theta = \alpha_\psi = 5.1$.



Синим цветом обозначена заданная траектория; красным цветом – траектория движения квадрокоптера; зеленым цветом – проекция траектории движения квадрокоптера на плоскость XY.

Рисунок 6 – Результат испытания контроллера траекторного управления



Проекции на плоскость XY заданной траектории (синим цветом) и фактической траектории движения квадрокоптера (красным цветом)

Рисунок 7 – Результат испытания контроллера траекторного управления

Среднеквадратичное отклонение в плоскости XY составило 0.11 м, по высоте – 0.019 м.

Испытания продемонстрировали уверенное удержание траектории квадрокоптером, что согласуется с результатами численного моделирования.

ЗАКЛЮЧЕНИЕ

В ходе работы был разработан и реализован программный модуль, основывающийся на подходе к организации вынужденного движения квадрокоптера в пространстве состояний на основе метода структурного синтеза. Данный программный модуль был использован для реализации системы управления движением по заданной траектории квадрокоптера с контроллером Pixhawk и бортовым ПО PX4, качество работы которой было подтверждено экспериментально.

Результаты работы представлены на 56-й Международной конференции МНСК-2018 в г. Новосибирске.

Выпускная квалификационная работа выполнена мной самостоятельно и с соблюдением правил профессиональной этики. Все использованные в работе материалы и заимствованные принципиальные положения (концепции) из опубликованной научной литературы и других источников имеют ссылки на них. Я несу ответственность за приведенные данные и сделанные выводы.

Я ознакомлен с программой государственной итоговой аттестации, согласно которой обнаружение плагиата, фальсификации данных и ложного цитирования является основанием для не допуска к защите выпускной квалификационной работы и выставления оценки «неудовлетворительно».

_____ _____

ФИО студента Подпись студента

« ____ » _____ 20__г.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ

1. Luukkonen T. Modelling and control of quadcopter, Independent research project in applied mathematics // Espoo: Aalto University, 2011. P. 26
2. Bouabdallah S., Siegwart R. Backstepping and sliding-mode techniques applied to an indoor micro quadrotor // Proc. of the 2005 IEEE Intern. Conf. on Robotics and Automation. 2005. P. 2259–2264.
3. Савицкий А. В., Павловский В. Е. Модель квадрокоптера и нейросетевой алгоритм управления // Препринты ИПМ им. М.В.Келдыша. 2017. № 77. 20 с.
4. Бойчук Л. М. Метод структурного синтеза нелинейных систем автоматического управления. М.: Энергия, 1971. 112 с.
5. Белоконь С. А., Золотухин Ю. Н., Мальцев А. С. и др. Управление параметрами полета квадрокоптера при движении по заданной траектории // Автометрия. 2012. 48, №5. С. 32-41.
6. Маргун А. А., Зименко К. А., Базылев Д. Н. и др. Система управления беспилотным летательным аппаратом, оснащенным робототехническим манипулятором // Научно-технический вестник информационных технологий, механики и оптики. 2014. №6 (94).
7. Pixhawk Flight Controller Hardware Project. [Электронный ресурс]. – Режим доступа: https://pixhawk.org/. – Дата обращения: 27.05.2018.
8. Open Source for Drones - PX4 Open Source Autopilot. [Электронный ресурс]. – Режим доступа: http://px4.io/. – Дата обращения: 27.05.2018.
9. ArduPilot Open Source Autopilot. [Электронный ресурс]. – Режим доступа: http://ardupilot.org/. – Дата обращения: 27.05.2018.
10. Dronecode - The Open Source UAV Platform. [Электронный ресурс]. – Режим доступа: https://www.dronecode.org/. – Дата обращения: 27.05.2018.
11. Flight Modes · PX4 Developer Guide. [Электронный ресурс]. – Режим доступа: https://dev.px4.io/en/concept/flight_modes.html. – Дата обращения: 27.05.2018.